Matematica delle Reti Neurali

Volume VIII

Strutture comuni e differenze fondamentali
tra Grandi Modelli Linguistici e AI Scientifica:

AlphaFold 2, AlphaFold 3, AlphaGenome, ESM-3
e i fondamenti matematici della modellazione biologica profonda

Temi trattati: Equivarianza SE(3) · Attenzione triangolare · Evoformer · IPA · FAPE · MSA · Tokenizzazione genomica multi-scala · Flash Attention · Gruppi di simmetria S_n vs SE(3) · AlphaGenome · Scientific Foundation Models · Diffusion per strutture molecolari · ESM-3 · Leggi di scala con inductive bias

Silvano Salvador

Continuazione dei Volumi I–VII (Sezioni 0–446)

Sezioni 447–510 · Derivazioni complete · Ogni simbolo definito · Confronti sistematici

*Nota metodologica: La notazione segue la Sezione 0 del Volume I. Ogni rimando a sezioni precedenti è esplicito nella forma (*Vol. I, §N*) o (*Vol. IV, §242*). Nessun prerequisito oltre al Volume I è assunto.*

Indice

Prefazione **6**

1 Sezione A — Il Substrato Condiviso **8**
1.1 §447 Il meccanismo di attenzione scalata: derivazione completa 8
1.1.1 Flash Attention: riformulazione per efficienza di memoria 8
1.1.2 Multi-Query e Grouped-Query Attention 9
1.2 §448 Positional encoding: tre filosofie e analisi comparativa 9
1.2.1 Encoding sinusoidale (Vaswani et al. 2017) 10
1.2.2 RoPE — Rotary Position Embedding (Su et al. 2024) 10
1.2.3 ALiBi (Press et al. 2022) 10
1.3 §449 Normalizzazione e blocchi feed-forward 11
1.3.1 Pre-LN vs Post-LN 11
1.3.2 SwiGLU e varianti gated FFN 11
1.4 §450 Modellazione autoregressiva del linguaggio e causal mask 11
1.5 §451 Leggi di scala per i LLM: derivazione Chinchilla 12
1.5.1 Legge di Kaplan (2020) 12
1.5.2 Legge di Chinchilla: derivazione dal problema di ottimizzazione 12
1.6 §452 Ottimizzazione: Adam e varianti moderne 13
1.6.1 AdamW e weight decay 13
1.6.2 Adafactor 13
1.6.3 Muon: aggiornamento ortogonalizzato 13
1.7 §453–455 Tabella sistematica del substrato comune 14
1.8 §454 Esempio numerico: calcolo esplicito dell'attenzione scalata 16
1.9 §455a Tokenizzazione: BPE, amino acidi e nucleotidi 16
1.9.1 Byte-Pair Encoding (BPE) nei LLM 16
1.9.2 Tokenizzazione aminoacidica in AlphaFold 17
1.9.3 Tokenizzazione nucleotidica in AlphaGenome 17
1.10 §455b Analisi del flusso del gradiente negli strati Transformer 18
1.10.1 Teorema di propagazione del gradiente con Pre-LN 18
1.10.2 Warm-up del learning rate: motivazione matematica 18
1.11 §455c Mixture of Experts: routing e bilanciamento del carico 18
1.11.1 Architettura formale 19
1.11.2 Il problema del load balancing 19
1.11.3 MoE in AI scientifica 20
1.12 §455d Speculative Decoding: inferenza accelerata con drafting 20
1.12.1 Algoritmo e correttezza 20
1.13 §455e RMSNorm: derivazione e confronto con LayerNorm 21

2 Sezione B — Simmetria, Gruppi e Equivarianza **21**
2.1 §456 Perché la simmetria è fondamentale: teoria del gruppo di simmetria . . . 21
2.2 §457 Il gruppo S_N e l'attenzione linguistica 22
2.3 §458 Il gruppo SE(3) e la struttura proteica 22
2.3.1 Perché AlphaFold NON usa armoniche sferiche 23
2.4 §459 Confronto algebrico: S_N vs SE(3) 23
2.5 §460 Frames rigidi e rappresentazione del backbone 23
2.6 §461–464 Costruire funzioni SE(3)-invarianti: dal primo principio 24
2.6.1 Invarianti scalari di primo ordine 25
2.6.2 Teorema di completezza degli invarianti 25
2.6.3 Costruzione delle features di coppia iniziali in AlphaFold 25

2.6.4 Il numero minimo di invarianti necessari . . . 25
2.6.5 La biforcazione tra S_N e SE(3) come teorema . . . 26
2.7 §462 Equivariant Graph Neural Networks: costruzione esplicita . . . 27
2.7.1 Messaggio equivariante . . . 27
2.7.2 Confronto con le architetture basate su armoniche sferiche . . . 27
2.8 §463 Dalla distanza alle feature equivarianti: SchNet, DimeNet, PaiNN . . . 28
2.8.1 SchNet (Schütt et al., 2017) . . . 28
2.8.2 DimeNet (Klicpera et al., 2020) . . . 28
2.8.3 PaiNN (Schütt et al., 2021) . . . 28
2.9 §464 Come l'IPA di AlphaFold si inserisce nel framework equivariante . . . 29
2.10 §464b Vantaggio in complessità campionaria dei modelli equivarianti . . . 29
2.11 §464c Espressività delle GNN: la gerarchia Weisfeiler-Lehman . . . 30
2.11.1 Il test di isomorfismo WL . . . 30

3 Sezione C — AlphaFold 2: Architettura Matematica Completa 31
3.1 §465 Panoramica architetturale e notazione . . . 31
3.2 §466 Multiple Sequence Alignment come memoria evolutiva . . . 31
3.3 §467 Attenzione assiale nell'Evoformer . . . 32
3.3.1 MSA row-wise attention con pair bias . . . 32
3.3.2 MSA column-wise attention . . . 32
3.3.3 Outer product mean . . . 32
3.4 §468–471 Aggiornamenti triangolari: la matematica dell'invariante geometrico 32
3.4.1 Aggiornamento moltiplicativo outgoing . . . 32
3.4.2 Aggiornamento moltiplicativo incoming . . . 33
3.4.3 Interpretazione geometrica . . . 33
3.4.4 Attenzione triangolare (starting node) . . . 33
3.4.5 Attenzione triangolare (ending node) . . . 33
3.5 §472–475 Lo Structure Module . . . 33
3.5.1 Aggiornamento del backbone con quaternioni . . . 33
3.6 §476 Invariant Point Attention (IPA): derivazione completa . . . 34
3.7 §477 Angoli torsionali e ricostruzione atomistica . . . 34
3.8 §478 La loss FAPE . . . 34
3.9 §479 pLDDT, pTM e recycling . . . 35
3.9.1 pLDDT: confidenza per residuo . . . 35
3.9.2 pTM score: confidenza globale . . . 35
3.9.3 Recycling come punto fisso . . . 35
3.10 §480 AlphaFold 3: diffusione su strutture molecolari . . . 35
3.10.1 Il Pairformer . . . 35
3.10.2 Loss di diffusione condizionale . . . 36
3.11 §472a Backbone update con quaternioni: derivazione completa . . . 36
3.11.1 Algebra dei quaternioni . . . 36
3.11.2 L'aggiornamento iterativo nel Structure Module . . . 37
3.11.3 Violazione dei legami e stop-gradient . . . 37
3.12 §473a Distillazione e training su strutture predette . . . 37
3.13 §474b Il diagramma di Ramachandran: vincoli geometrici del backbone . . . 38
3.13.1 Definizione degli angoli di Ramachandran . . . 38
3.13.2 La distribuzione di Ramachandran come prior fisico . . . 38
3.13.3 Confronto con i LLM . . . 39
3.14 §480b Score Matching e la connessione con i modelli di diffusione in AF3 . . . 39
3.14.1 Score function e stima . . . 39
3.14.2 Denoising Score Matching (DSM) e DDPM . . . 39
3.14.3 Il processo di campionamento: DDPM vs DDIM . . . 40

3.15 §480c Flow Matching: un'alternativa più stabile alla diffusione . . . 40
3.15.1 Flussi di probabilità condizionali . . . 40
3.15.2 Conditional Flow Matching (CFM) . . . 40

4 Sezione D — AlphaGenome: La Frontiera Genomica 41
4.1 §481 Motivazione e dominio . . . 41
4.2 §482 Tokenizzazione genomica multi-scala . . . 41
4.2.1 Tokenizzazione al livello di nucleotide . . . 42
4.2.2 Stem convolutivo multi-scala . . . 42
4.2.3 Livelli di risoluzione multipla . . . 42
4.3 §483–486 Attenzione cross-scala . . . 42
4.3.1 Position encoding genomico . . . 42
4.4 §487–490 Complementarità di Watson-Crick come simmetria . . . 43
4.4.1 Formalizzazione matematica . . . 43
4.4.2 Implementazione tramite weight sharing e data augmentation . . . 43
4.5 §491–494 Teste di predizione multi-task . . . 44
4.5.1 Head per l'espressione genica (RNA-seq) . . . 44
4.5.2 Head per l'accessibilità della cromatina (ATAC-seq) . . . 44
4.5.3 Head per la struttura 3D del cromosoma (Hi-C) . . . 44
4.5.4 Head per i siti di legame dei fattori di trascrizione (ChIP-seq) . . . 44
4.5.5 Head per lo splicing alternativo . . . 45
4.6 §494 Funzione di perdita multi-task . . . 45
4.7 §495 Confronto matematico AlphaFold vs AlphaGenome . . . 45
4.8 §492a Dinamiche del gradiente nel training multi-task . . . 46
4.8.1 Conflitto di gradiente . . . 46
4.8.2 PCGrad: proiezione per ridurre i conflitti . . . 47
4.9 §493 Overfitting e regolarizzazione nel dominio genomico . . . 47
4.9.1 Augmentation specifica al DNA . . . 47
4.9.2 Validazione su contigs genomici separati . . . 47
4.10 §494b Attenzione dilatata e gerarchica per sequenze genomiche . . . 48
4.10.1 Attenzione a finestre locali con dilatazione . . . 48
4.10.2 Confronto con l'attenzione cross-scala di AlphaGenome . . . 48
4.11 §495b Metilazione del DNA e segnali epigenetici come feature aggiuntive . . . 48
4.11.1 La metilazione del DNA . . . 48
4.11.2 Modifiche degli istoni e chromatin accessibility . . . 49
4.12 §495c Quantificazione dell'incertezza nelle predizioni genomiche . . . 49
4.12.1 Conformal Prediction per segnali genomici . . . 49

5 Sezione E — Mappa Sistematica delle Differenze Matematiche 50
5.1 §496 La biforcazione fondamentale: derivazione formale . . . 50
5.2 §497–500 Tabella comparativa a 15 dimensioni . . . 50
5.3 §501 La rappresentazione doppia: dimostrazione di necessità . . . 51
5.4 §502 Confronto delle loss: dalla bit-entropia alla geometria . . . 52
5.4.1 Cross-entropy (LLM) . . . 52
5.4.2 FAPE (AlphaFold 2) . . . 52
5.4.3 Pearson correlation loss (AlphaGenome) . . . 52
5.5 §503 pLDDT vs confidenza nei LLM . . . 52
5.5.1 pLDDT in AlphaFold 2 . . . 52
5.5.2 Confidenza nei LLM: token probability e perplexity . . . 53
5.6 §504 Recycling vs Chain-of-Thought: due forme di iterazione . . . 53
5.7 §500a Implicazioni computazionali della biforcazione: analisi della complessità 54
5.7.1 Complessità dell'inferenza . . . 54

5.7.2 Analisi della finestra di contesto effettiva . . . 54
5.8 §500b Analisi della generalization: teoria PAC . . . 55
5.9 §500c Complessità di Rademacher e generalizzazione comparata . . . 55
5.9.1 Stima della complessità di Rademacher per Transformer . . . 55
5.9.2 Implicazione per AlphaFold . . . 56
5.10 §503b Calibrazione dei modelli: temperature scaling, Platt, isotonica . . . 56
5.10.1 Expected Calibration Error (ECE) . . . 56
5.10.2 Temperature Scaling . . . 56
5.11 §504b Prospettiva dell'informazione teorica: LLM vs AI scientifica . . . 56
5.11.1 Minimum Description Length e architettura . . . 57
5.11.2 Mutua informazione tra sequenza e struttura/funzione . . . 57

6 Sezione F — Convergenza Futura: Scientific Foundation Models 58
6.1 §505 Il continuum LLM–AI scientifica . . . 58
6.2 §506 ESM-3: convergenza in atto . . . 58
6.2.1 Architettura di ESM-3 . . . 58
6.2.2 Quantizzazione strutturale con VQ-VAE . . . 58
6.3 §507 Scientific Foundation Models: architettura unificata . . . 59
6.4 §508 Il Neural Scaling per l'AI scientifica . . . 59
6.4.1 Scaling dei protein language model . . . 59
6.4.2 Scaling laws con inductive bias: teoria . . . 60
6.5 §509 La matematica dell'emergenza nei Scientific Foundation Models . . . 60
6.5.1 L'ipotesi dello spazio biologico universale . . . 60
6.5.2 Evidenza empirica . . . 60
6.6 §510 Sintesi finale e prospettive . . . 61
6.7 §506b State Space Models e Mamba per sequenze biologiche lunghissime . . . 61
6.7.1 SSM lineari e la connessione con i sistemi dinamici . . . 62
6.7.2 Mamba: SSM selettivo con gating . . . 62
6.7.3 Confronto computazionale: Mamba vs Transformer . . . 62
6.8 §507b Apprendimento contrastivo cross-modale per la biologia . . . 63
6.8.1 Perdita InfoNCE per allineamento cross-modale biologico . . . 63
6.8.2 Applicazioni: ProteinCLIP e GenomeCLIP . . . 63
6.9 §509b Problemi aperti: la frontiera matematica dell'AI scientifica . . . 63
6.9.1 1. Generalizzazione fuori-distribuzione in biologia strutturale . . . 63
6.9.2 2. Interpretabilità geometrica delle rappresentazioni latenti . . . 64
6.9.3 3. Leggi di scala con dati strutturali aumentati . . . 64
6.9.4 4. Convergenza cross-dominio: verso la matematica del biosistema unificato . . . 64

A Appendice A — Glossario dei Simboli 65

B Appendice B — Note Storiche e Riferimenti 66

C Appendice C — Derivazioni Complete 67
C.1 C.1 Derivazione completa degli aggiornamenti triangolari dell'Evoformer . . . 67
C.1.1 Motivazione geometrica . . . 67
C.1.2 Aggiornamento moltiplicativo outgoing (forma completa) . . . 67
C.1.3 Aggiornamento moltiplicativo incoming . . . 68
C.1.4 Attenzione triangolare con starting node (forma completa) . . . 68
C.1.5 Attenzione triangolare con ending node . . . 68
C.2 C.2 Derivazione del gradiente di Adam con bias correction . . . 68

D Appendice D — Raccolta degli Algoritmi **70**
D.1 D.1 Blocco Evoformer completo . . . 70
D.2 D.2 Algoritmo IPA completo . . . 70
D.3 D.3 Generazione autoregressiva con KV-cache . . . 70

E Appendice E — Schede Riassuntive dei Modelli **70**

F Appendice F — Geometria Differenziale di SE(3) **70**
F.1 F.1 Struttura di gruppo di Lie . . . 70
F.1.1 La mappa esponenziale . . . 70
F.1.2 Geodetiche e distanza di Riemanniana . . . 71
F.2 F.2 Interpolazione SLERP per rotazioni . . . 72
F.3 F.3 Misura di Haar su SO(3) . . . 72

G Appendice G — Connessioni con la Teoria delle Informazioni Biologica **72**
G.1 G.1 Capacità dei canali biologici . . . 72
G.2 G.2 Entropia evolutiva e MSA . . . 72
G.3 G.3 La compressione dell'informazione strutturale . . . 73

Bibliografia **74**

Prefazione al Volume VIII

I sette volumi precedenti di questa serie hanno costruito un panorama matematico eccezionalmente ampio: dalle fondamenta del neurone artificiale (Vol. I) fino ai Grandi Modelli Linguistici, al loro allineamento (Vol. VII), passando per la geometria differenziale dell'informazione, la causalità, le leggi di scala, e un'ampia trattazione di AlphaFold 2 nel contesto della biologia computazionale (Vol. IV, §§242–246).

Tuttavia, un angolo prospettico era rimasto sistematicamente scoperto: nessun volume aveva mai assunto la postura *comparativa e contrastiva* — ossia non si era mai chiesto che cosa *distingue* matematicamente un Large Language Model da un sistema di AI scientifica come AlphaFold, né che cosa i due *condividono* a un livello di struttura più profondo di quello delle singole architetture.

Questo volume colma precisamente quella lacuna, con un'ambizione maggiore rispetto alla prima stesura: non solo catalogare le differenze, ma *derivare* da principi primi ogni struttura matematica, mostrare come le scelte architetturali siano matematicamente *necessarie* dati i vincoli del dominio fisico, ed estendere l'analisi ai modelli più recenti — AlphaFold 3, ESM-3, Evo-2 — che hanno ridefinito i confini del campo.

Tesi centrale del Volume VIII

L'attenzione è il vocabolario *comune* di LLM e AI scientifica; ma la geometria dell'oggetto modellato — scalare-sequenziale per il linguaggio, euclidea-tridimensionale per le proteine, multi-scala per il genoma — determina *interamente* la matematica che viene costruita sopra di essa. Ogni differenza architetturale osservata tra un LLM e un sistema come AlphaFold è un *teorema*, non una scelta di design.

Struttura del volume

Il testo è organizzato in sei sezioni tematiche (§§447–510), ognuna espansa rispetto alla prima stesura per includere derivazioni complete, esempi numerici, e connessioni con i modelli più recenti:

- **Sezione A** (§§447–455): *Il substrato condiviso* — attenzione scalata con derivazione della varianza, Flash Attention, Multi-Query e Grouped-Query Attention, positional encoding, layer normalization, Adam e varianti moderne (AdamW, Adafactor, Muon), scaling laws con derivazione Chinchilla.
- **Sezione B** (§§456–464): *Simmetria e gruppi* — teoria dei gruppi di Lie, rappresentazione di S_N e SE(3), costruzione di reti equivarianti dal primo principio, armoniche sferiche e network $E(n)$-equivarianti, confronto algebrico sistematico.
- **Sezione C** (§§465–480): *AlphaFold 2 in profondità* — MSA completo con covarianza evolutiva, Evoformer con tutti e quattro gli aggiornamenti triangolari derivati, IPA con dimostrazione completa dell'invarianza, FAPE con gradiente esplicito, pLDDT, pTM score, recycling come punto fisso, distillation.
- **Sezione C2** (§§479–480): *AlphaFold 3* — diffusione su strutture molecolari, Pairformer, loss di diffusione condizionale, estensione a ligandi e acidi nucleici.
- **Sezione D** (§§481–495): *AlphaGenome* — tokenizzazione genomica multi-scala, attenzione cross-scala, complementarità di Watson-Crick come simmetria, teste multi-task con derivazione delle loss, confronto con Enformer e Basenji.

- **Sezione E** (§§496–504): *Mappa sistematica delle differenze matematiche* — tabella a 15 dimensioni, proposizioni formali sulla necessità della rappresentazione di coppia, confronto delle loss, pLDDT vs pLPP, recycling vs chain-of-thought.
- **Sezione F** (§§505–510): *Convergenza futura* — Scientific Foundation Models, ESM-3 con VQ-VAE strutturale, leggi di scala con inductive bias, matematica dell'emergenza cross-modale, Evo-2, Chai-1.

Note per il lettore

Ogni sezione è autocontenuta nel senso che tutti i simboli vengono ridefiniti localmente quando introdotti per la prima volta in quel contesto, anche se già apparsi in sezioni precedenti. I rimandi ai volumi precedenti sono indicati esplicitamente e il lettore non è obbligato a consultarli per seguire le derivazioni. Le dimostrazioni complete sono indicate con il simbolo $\square$ e possono essere saltate in una prima lettura senza perdere il filo della trattazione.

1 Sezione A — Il Substrato Condiviso

1.1 §447 Il meccanismo di attenzione scalata: derivazione completa

L'attenzione scalata con prodotto scalare (*Scaled Dot-Product Attention*, Vaswani et al. 2017) è la struttura computazionale centrale sia nei Transformer linguistici sia nell'Evoformer di AlphaFold 2 e nella rete genomica di AlphaGenome. Ricaviamo la formula dall'inizio.

Sia $X \in \mathbb{R}^{N\times d}$ una matrice di input, dove N è la lunghezza della sequenza e d la dimensione dell'embedding. Tre proiezioni lineari producono le matrici di query, key e value:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V, \qquad W^Q, W^K, W^V \in \mathbb{R}^{d\times d_k}. \tag{1}$$

Motivazione del fattore $1/\sqrt{d_k}$. Il prodotto scalare $\langle Q_{i\cdot}, K_{j\cdot}\rangle = \sum_{\ell=1}^{d_k} Q_{i\ell}K_{j\ell}$. Se $Q_{i\ell}, K_{j\ell} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, allora ogni termine $Q_{i\ell}K_{j\ell}$ ha media 0 e varianza 1. La somma di d_k termini indipendenti ha varianza d_k, cioè deviazione standard $\sqrt{d_k}$.

Lemma 1.1 (Stabilizzazione della varianza)**.** *Sotto l'ipotesi $Q_{i\ell}, K_{j\ell} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$:*

$$\operatorname{Var}\left(\frac{\langle Q_{i\cdot}, K_{j\cdot}\rangle}{\sqrt{d_k}}\right) = 1.$$

Dimostrazione. $\operatorname{Var}(\langle Q_{i\cdot}, K_{j\cdot}\rangle) = \sum_\ell \operatorname{Var}(Q_{i\ell}K_{j\ell}) = \sum_\ell \mathbb{E}[Q_{i\ell}^2]\mathbb{E}[K_{j\ell}^2] = d_k$. Dividendo per $\sqrt{d_k}$ si riduce la varianza di un fattore d_k, portandola a 1. □ □

Senza il fattore $1/\sqrt{d_k}$, per d_k grande i punteggi avrebbero grandezza $O(\sqrt{d_k})$, spingendo il softmax in una zona di gradiente quasi nullo (zona *saturation*) e impedendo l'apprendimento.

La matrice di attenzione e l'output sono:

$$A = \operatorname{softmax}\left(\frac{QK^\mathsf{T}}{\sqrt{d_k}}\right) \in \mathbb{R}^{N\times N}, \qquad \operatorname{Attn}(Q,K,V) = AV \in \mathbb{R}^{N\times d_k}. \tag{2}$$

Il softmax agisce *per righe*: $A_{ij} = e^{s_{ij}}/\sum_k e^{s_{ik}}$ dove $s_{ij} = \langle Q_{i\cdot}, K_{j\cdot}\rangle/\sqrt{d_k}$.

L'attenzione multi-testa (*Multi-Head Attention*, MHA) esegue H attenzioni parallele con proiezioni indipendenti e le concatena:

$$\operatorname{MHA}(X) = \operatorname{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O, \qquad \text{head}_h = \operatorname{Attn}(XW_h^Q, XW_h^K, XW_h^V). \tag{3}$$

Il costo computazionale è $O(N^2 d)$ in tempo e $O(N^2)$ in memoria (per la matrice A).

1.1.1 Flash Attention: riformulazione per efficienza di memoria

La bottleneck della MHA standard non è il costo $O(N^2d)$ in FLOPs, ma la necessità di materializzare l'intera matrice $A \in \mathbb{R}^{N\times N}$ in memoria ad alta larghezza di banda (HBM). Per $N = 8192$ e $d = 128$, A occupa $8192^2 \times 4$ byte ≈ 268 MB per singola testa.

Flash Attention (Dao et al., 2022; Dao, 2023) risolve il problema con la *tiling*: si suddivide Q in blocchi $Q_{(1)}, \ldots, Q_{(T_r)}$ e K, V in blocchi $K_{(1)}, \ldots, K_{(T_c)}$ e si calcola il softmax in modo numericamente stabile senza mai materializzare A completa.

Definizione 1.2 (Safe softmax con running statistics)**.** Dato un vettore $s \in \mathbb{R}^N$, il softmax sicuro si calcola mantenendo:

$$m_j = \max_{1\le i\le j} s_i, \quad d_j = \sum_{i=1}^{j} e^{s_i - m_j}, \quad \text{output}_i = \frac{e^{s_i - m_N}}{d_N}.$$

Algorithm 1 Flash Attention v2 (schema semplificato)

Dividi Q in blocchi $Q_1, \dots, Q_{T_r}$, K, V in $K_1, \dots, K_{T_c}$
for $i = 1$ to T_r **do**
  Carica Q_i in SRAM
  Inizializza $O_i \leftarrow \mathbf{0}$, $\ell_i \leftarrow \mathbf{0}$, $m_i \leftarrow -\infty$
  for $j = 1$ to T_c **do**
    Carica K_j, V_j in SRAM
    $S_{ij} \leftarrow Q_i K_j^{\mathsf{T}} / \sqrt{d_k}$
    $\tilde{m}_{ij} \leftarrow \mathrm{rowmax}(S_{ij})$
    $m_i^{\mathrm{new}} \leftarrow \max(m_i, \tilde{m}_{ij})$
    $\ell_i^{\mathrm{new}} \leftarrow e^{m_i - m_i^{\mathrm{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\mathrm{new}}} \mathrm{rowsum}(e^{S_{ij} - \tilde{m}_{ij}})$
    $O_i \leftarrow \frac{e^{m_i - m_i^{\mathrm{new}}} \ell_i \cdot O_i + e^{\tilde{m}_{ij} - m_i^{\mathrm{new}}} e^{S_{ij} - \tilde{m}_{ij}} V_j}{\ell_i^{\mathrm{new}}}$
    $(m_i, \ell_i) \leftarrow (m_i^{\mathrm{new}}, \ell_i^{\mathrm{new}})$
  end for
  Scrivi O_i in HBM
end for

Flash Attention generalizza questa procedura al caso matriciale: per ogni blocco (i, j) si aggiornano le statistiche (m, ℓ, O) locali e poi si correggono moltiplicando per il fattore $e^{m_{\mathrm{old}} - m_{\mathrm{new}}}$.

Proposizione 1.3 (Complessità Flash Attention)**.** *Flash Attention calcola l'identico output di MHA standard con:*

- *complessità in memoria $O(N)$ (invece di $O(N^2)$);*
- *complessità in FLOPs $O(N^2 d)$ (identica, con costante nascosta minore grazie alla coalescenza degli accessi alla memoria).*

1.1.2 Multi-Query e Grouped-Query Attention

Multi-Query Attention (MQA, Shazeer 2019) condivide *un'unica* coppia (K, V) tra tutte le teste:

$$\mathrm{head}_h = \mathrm{Attn}(XW_h^Q,\ XW^K,\ XW^V), \qquad h = 1, \dots, H.$$

Riduce la memoria della KV-cache da $O(H \cdot N \cdot d_k)$ a $O(N \cdot d_k)$, cruciale per l'inferenza con sequenze lunghe.

Grouped-Query Attention (GQA, Ainslie et al. 2023) generalizza MQA partizionando le H teste in G gruppi ($G < H$), con una coppia (K_g, V_g) condivisa all'interno del gruppo g:

$$\mathrm{head}_h = \mathrm{Attn}(XW_h^Q,\ XW_{g(h)}^K,\ XW_{g(h)}^V),$$

dove $g(h) = \lceil hG/H \rceil$. GQA interpola tra MHA ($G = H$) e MQA ($G = 1$), ed è adottata da LLaMA-3, Gemma, Mistral.

Nota. AlphaFold 2 e AlphaGenome usano un numero fisso di teste ($H = 8$ nell'Evoformer) senza MQA/GQA, perché la lunghezza delle sequenze proteiche ($L \leq 2500$) è abbastanza piccola da non richiedere ottimizzazioni aggressive della KV-cache. Al contrario, per il genoma AlphaGenome usa GQA con $G = 4$ nelle scale più fini dove $N > 8000$.

1.2 §448 Positional encoding: tre filosofie e analisi comparativa

Il meccanismo di attenzione in (2) è *permutation-equivariante*: $\mathrm{Attn}(\pi X) = \pi\, \mathrm{Attn}(X)$ per qualsiasi permutazione π delle righe. Questo è un vantaggio (non assume ordine a priori)

e un limite (perde l'informazione posizionale). Nei LLM si inietta la posizione in tre modi fondamentali.

1.2.1 Encoding sinusoidale (Vaswani et al. 2017)

Per la posizione p e la dimensione $2i/2i+1$:

$$\mathrm{PE}(p, 2i) = \sin\!\Big(\frac{p}{10000^{2i/d}}\Big), \quad \mathrm{PE}(p, 2i+1) = \cos\!\Big(\frac{p}{10000^{2i/d}}\Big). \tag{4}$$

La scelta trigonometrica non è arbitraria. Sia $\omega_i = 10000^{-2i/d}$. La propriet chiave è:

$$\mathrm{PE}(p+k) = M_k\,\mathrm{PE}(p), \qquad M_k \in \mathbb{R}^{d\times d}, \tag{5}$$

dove M_k è una matrice di rotazioni a blocchi 2×2 con blocco i-esimo $\begin{pmatrix}\cos(k\omega_i) & -\sin(k\omega_i)\\ \sin(k\omega_i) & \cos(k\omega_i)\end{pmatrix}$. Questa linearità permette al modello di calcolare distanze relative con una semplice trasformazione lineare — il motivo per cui sinusoidal PE generalizza a lunghezze non viste in training.

1.2.2 RoPE — Rotary Position Embedding (Su et al. 2024)

RoPE incorpora la posizione direttamente nel prodotto scalare tra query e key. Per $d_k = 2$ la rotazione è:

$$R_p = \begin{pmatrix}\cos p\theta & -\sin p\theta\\ \sin p\theta & \cos p\theta\end{pmatrix}, \quad \theta \in \mathbb{R}. \tag{6}$$

Per d_k generico, si applicano $d_k/2$ rotazioni 2D indipendenti con frequenze $\theta_i = 10000^{-2i/d_k}$.

Proposizione 1.4 (Proprietà di relatività di RoPE)**.** *Il prodotto scalare tra la query alla posizione p e la key alla posizione q dipende solo dalla differenza relativa $p-q$:*

$$\langle R_p q, R_q k\rangle = \langle R_{p-q} q, k\rangle. \tag{7}$$

Dimostrazione. $\langle R_p q, R_q k\rangle = q^\top R_p^\top R_q k = q^\top R_{q-p} k = \langle q, R_{q-p} k\rangle = \langle R_{p-q} q, k\rangle$, dove si è usato $R_p^\top = R_{-p}$ e $R_a R_b = R_{a+b}$. □ □

RoPE è oggi lo standard de facto: LLaMA 2/3, Gemma, Qwen, Mistral, GPT-4 (presunto) lo adottano. La sua variante *YaRN* (Peng et al., 2023) modifica le frequenze θ_i per estendere la finestra di contesto oltre quella di training usando interpolazione e NTK-awareness.

1.2.3 ALiBi (Press et al. 2022)

ALiBi aggiunge un bias lineare negativo proporzionale alla distanza al punteggio di attenzione:

$$s_{ij}^{\mathrm{ALiBi}} = \frac{\langle Q_{i\cdot}, K_{j\cdot}\rangle}{\sqrt{d_k}} - m_h|i-j|, \tag{8}$$

dove $m_h > 0$ è una pendenza specifica per testa h, tipicamente $m_h = 2^{-8h/H}$. L'intuizione è che attenzioni lontane ricevono una penalità crescente, inducendo un inductive bias di *località* senza modificare i pesi.

Confronto chiave: position encoding in LLM vs AlphaFold

Nei LLM il position encoding è un'aggiunta *esterna* alla meccanica dell'attenzione, necessaria perché la sequenza linguistica è ordinata ma l'attenzione è invariante alla permutazione. In AlphaFold la posizione nella catena amminoacidica è una feature

discreta aggiunta all'embedding, ma la struttura 3D non ha alcun ordinamento canonico: è la geometria fisica a imporre i vincoli, non una sequenza. In AlphaGenome, la posizione genomica ha una struttura logaritmica (distanze da 1 bp a 1 Mbp) che richiede un encoding specializzato (§483).

1.3 §449 Normalizzazione e blocchi feed-forward

Il blocco Transformer (sia LLM sia AlphaFold/AlphaGenome) alterna attenzione e rete feed-forward con residual connection e layer normalization. La struttura è:

$$h^{(l+\frac{1}{2})} = \mathrm{LayerNorm}\Big(h^{(l)} + \mathrm{MHA}\Big(h^{(l)}\Big)\Big), \tag{9}$$

$$h^{(l+1)} = \mathrm{LayerNorm}\Big(h^{(l+\frac{1}{2})} + \mathrm{FFN}\Big(h^{(l+\frac{1}{2})}\Big)\Big), \tag{10}$$

con $\mathrm{FFN}(x) = W_2 \max(0, W_1 x + b_1) + b_2$ (ReLU) oppure $W_2\,\mathrm{GELU}(W_1 x + b_1) + b_2$ nelle varianti più recenti.

La layer normalization è:

$$\mathrm{LayerNorm}(x) = \gamma \odot \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \quad \mu = \frac{1}{d}\sum_i x_i, \quad \sigma^2 = \frac{1}{d}\sum_i (x_i - \mu)^2, \tag{11}$$

dove $\gamma, \beta \in \mathbb{R}^d$ sono parametri apprendibili e $\varepsilon \sim 10^{-5}$ previene la divisione per zero.

1.3.1 Pre-LN vs Post-LN

La formulazione originale di Vaswani et al. usa Post-LN (normalizzazione *dopo* la residual connection). La variante Pre-LN, adottata da GPT-3 e dai modelli moderni, normalizza *prima*:

$$h^{(l+\frac{1}{2})} = h^{(l)} + \mathrm{MHA}\Big(\mathrm{LayerNorm}(h^{(l)})\Big).$$

Proposizione 1.5 (Stabilità del gradiente in Pre-LN)**.** *Con Pre-LN, il gradiente del segnale diretto lungo il percorso residuale è $\partial h^{(L)}/\partial h^{(0)} = I + \sum_l \Delta_l$, dove i termini Δ_l sono la Jacobiana del blocco l. Poiché la LayerNorm stabilizza la scala, $\|\Delta_l\|$ rimane controllata in ogni strato, prevenendo il vanishing/exploding gradient senza warmup lungo.*

1.3.2 SwiGLU e varianti gated FFN

I modelli recenti (LLaMA 2/3, Gemma, Mistral) sostituiscono la FFN con una variante *gated*:

$$\mathrm{SwiGLU}(x) = (\mathrm{Swish}(W_1 x)) \odot (W_3 x)\, W_2, \tag{12}$$

dove $\mathrm{Swish}(z) = z \cdot \sigma(z)$ e $\odot$ è il prodotto di Hadamard. SwiGLU utilizza tre matrici di proiezione invece di due, ma riduce la dimensione nascosta da $4d$ a $\frac{2}{3} \cdot 4d = \frac{8d}{3}$ per mantenere lo stesso numero di parametri. AlphaFold 2 usa la FFN standard con ReLU, mentre AlphaGenome usa GELU.

1.4 §450 Modellazione autoregressiva del linguaggio e causal mask

Un LLM modella la distribuzione di probabilità di una sequenza di token $x = (x_1, x_2, \ldots, x_T)$ via regola della catena:

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1};\, \theta). \tag{13}$$

L'obiettivo di training è massimizzare la log-verosimiglianza:

$$\mathcal{L}_{\mathrm{LM}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}). \tag{14}$$

La causal mask. La causalità è implementata imponendo $A_{ij} = 0$ per $j > i$, cioè azzerando i punteggi con $s_{ij} = -\infty$ per $j > i$ prima del softmax. In forma matriciale, il punteggio mascherato è:

$$S_{ij}^{\text{masked}} = \begin{cases} s_{ij} & \text{se } j \leq i \\ -\infty & \text{se } j > i \end{cases}. \tag{15}$$

Nota. AlphaFold 2 **non usa** causal mask. L'informazione strutturale di una proteina non ha causalità temporale: tutti i residui interagiscono con tutti. La scelta dell'attenzione bidirezionale completa è quindi non solo lecita ma *necessaria* per la corretta modellazione delle interazioni a lungo raggio nella struttura 3D. AlphaGenome, modellando una sequenza di DNA con contesto biologico bidirezionale, usa ugualmente attenzione bidirezionale.

KV-Cache. Durante la generazione autoregressiva, si ricalcolano a ogni passo le chiavi e i valori solo per il token nuovo, conservando $(K_{<t}, V_{<t})$ in cache. La complessità della generazione di T token è $O(T^2 d)$ senza cache e $O(Td)$ per passo con cache (costo totale $O(T^2 d)$ uguale, ma latenza per token $O(Td)$ invece di $O(T^2 d)$).

1.5 §451 Leggi di scala per i LLM: derivazione Chinchilla

Le leggi di scala (*scaling laws*) quantificano come le prestazioni di un LLM variano con il numero di parametri N, la dimensione del dataset D (in token), e il compute $C \approx 6ND$ (FLOPs, approssimazione della regola dei parametri).

1.5.1 Legge di Kaplan (2020)

Kaplan et al. derivano empiricamente, fissando D e variando N:

$$L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha}, \quad \alpha \approx 0.076, \quad N_c \approx 8.8 \times 10^{13}. \tag{16}$$

con L la cross-entropy sui dati di test. La legge di potenza suggerisce che raddoppiare N riduce L di un fattore $2^{-\alpha} \approx 0.949$, cioè circa il 5%.

1.5.2 Legge di Chinchilla: derivazione dal problema di ottimizzazione

Hoffmann et al. (2022) considerano una loss parametrizzata da entrambi N e D:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}, \tag{17}$$

dove E è la perdita irriducibile (entropia del dataset), A/N^{α} cattura il miglioramento con più parametri, e B/D^{β} cattura il miglioramento con più dati.

Il problema di ottimizzazione vincolata è:

$$\min_{N,D} L(N, D) \quad \text{s.t.} \quad C = 6ND. \tag{18}$$

Usando la condizione KKT (o semplicemente sostituendo $D = C/(6N)$ e derivando):

$$\frac{\partial}{\partial N}\left[E + \frac{A}{N^{\alpha}} + \frac{B \cdot 6N^{\beta}}{C^{\beta}}\right] = 0 \tag{19}$$

$$-\alpha \frac{A}{N^{\alpha+1}} + \frac{\beta B (6)^{\beta}}{C^{\beta}} N^{\beta-1} = 0 \tag{20}$$

$$N^{\alpha+\beta} = \frac{\alpha A C^{\beta}}{\beta B (6)^{\beta}} \tag{21}$$

Con i valori sperimentali $\alpha \approx 0.34$, $\beta \approx 0.28$, $A \approx 406.4$, $B \approx 410.7$, si ottiene:

$$N^* \propto C^{0.5}, \qquad D^* \propto C^{0.5}. \tag{22}$$

La regola di Chinchilla afferma che il modello ottimale ha $D^* \approx 20N^*$ token di training.

Scaling laws nell'AI scientifica

Non esiste un equivalente diretto delle scaling laws per AlphaFold: le prestazioni (TM-score, GDT) dipendono in modo non-monotono dalla complessità della proteina target, e l'architettura incorpora inductive bias geometrico che "vale" come enormi quantità di dati aggiuntivi (cfr. §508). AlphaGenome mostra invece leggi di scala più simili agli LLM per i task di predizione genomica, suggerendo che la modellazione di sequenze discrete obbedisce a regolarità universali indipendenti dal dominio.

1.6 §452 Ottimizzazione: Adam e varianti moderne

Sia θ il vettore di tutti i parametri e $g_t = \nabla_\theta \mathcal{L}(\theta_t)$ il gradiente al passo t. L'ottimizzatore Adam (Kingma e Ba, 2014) mantiene le stime dei primi due momenti:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{23}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{24}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{25}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}. \tag{26}$$

1.6.1 AdamW e weight decay

AdamW (Loshchilov e Hutter, 2019) separa il weight decay λ dall'aggiornamento adattivo:

$$\theta_{t+1} = \theta_t - \alpha \left[\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_t \right]. \tag{27}$$

La differenza con la regolarizzazione ℓ_2 classica (che aggiunge $\lambda\theta$ a g_t prima di Adam) è che AdamW applica il decay *dopo* la riscalatura adattiva, producendo un decay uniforme indipendente dalla magnitudine del gradiente.

1.6.2 Adafactor

Adafactor (Shazeer e Stern, 2018) riduce la memoria di Adam da $O(|\theta|)$ a $O(\sqrt{|\theta|})$ fattorizzando v_t per le matrici di pesi:

$$V_t = r_t c_t^\top / \mathbf{1}^\top r_t, \quad r_{t,i} = \sum_j (G_t)_{ij}^2, \quad c_{t,j} = \sum_i (G_t)_{ij}^2 \cdot \frac{1}{r_{t,i}},$$

dove G_t è il gradiente matriciale. La matrice V_t è approssimata come prodotto esterno $r_t c_t^\top$ normalizzato, riducendo il costo da $O(m \cdot n)$ a $O(m+n)$.

1.6.3 Muon: aggiornamento ortogonalizzato

Muon (Jordan et al., 2024) è un ottimizzatore basato su *Nesterov Momentum* con un passo di ortogonalizzazione della matrice di aggiornamento:

$$G_t^{\text{orth}} = \text{NS}(G_t), \qquad \theta_{t+1} = \theta_t - \alpha G_t^{\text{orth}}, \tag{28}$$

dove $\mathrm{NS}(\cdot)$ denota l'iterazione di Newton-Schulz per calcolare la matrice con colonne ortonormali più vicina a G_t:

$$X_0 = G_t/\|G_t\|_F, \quad X_{k+1} = \frac{3}{2}X_k - \frac{1}{2}X_k X_k^\top X_k.$$

Muon è motivato dalla teoria dell'ottimizzazione su varietà Riemanniane: l'aggiornamento ortogonale rispetta la metrica naturale dello spazio di Stiefel delle matrici a colonne ortonormali, producendo stime del gradiente con norma spettrale controllata.

Tabella 1: Confronto degli ottimizzatori usati da LLM e AI scientifica

Ottimizzatore	Memoria	d_k-invariante	Decay	Usato in
Adam	$O(2\|\theta\|)$	No (via eps)	ℓ_2 aggiunto	GPT-3, BERT
AdamW	$O(2\|\theta\|)$	No	ℓ_2 separato	LLaMA, Gemma, AF2
Adafactor	$O(\sqrt{\|\theta\|})$	Sì	Opzionale	T5, PaLM
Muon	$O(\|\theta\|)$	Sì (ortonorm.)	Opzionale	Modelli recenti

AlphaFold 2 utilizza Adam con $\beta_1 = 0.9$, $\beta_2 = 0.999$, gradient clipping a norma 0.1, e un learning rate schedule con warm-up lineare (10^4 passi) seguito da decadimento esponenziale. I valori degli iperparametri non differiscono sostanzialmente da quelli dei LLM: questa è una delle evidenze più forti della tesi di condivisione del substrato di ottimizzazione.

1.7 §453–455 Tabella sistematica del substrato comune

Tabella 2: Strutture matematiche condivise tra LLM, AlphaFold 2 e AlphaGenome

Componente	**LLM**	**AlphaFold / AlphaGenome**	**Identico?**
Attenzione scalata	eq. (2)	eq. (2) (con bias aggiuntivi)	$\approx$
Multi-head attention	eq. (3)	Varianti assiali (§467)	$\approx$
Flash Attention	Sì (inferenza)	AlphaFold 3 sì; AF2 no	$\approx$
Layer normalization	eq. (11)	Identica	✓
Pre-LN / Post-LN	Pre-LN moderno	Post-LN (AF2); Pre-LN (AG)	$\approx$
Residual connection	eq. (9)	Identica	✓
Feed-forward block	SwiGLU / ReLU	ReLU (AF2), GELU (AG)	$\approx$
Ottimizzatore	AdamW	Adam (AF2), AdamW (AG)	$\approx$
Embedding di input	Lookup table	Lookup table	✓
Gradient clipping	Sì	Sì	✓
LR schedule	Cosine / linear	Linear warmup + exp	$\approx$
Position encoding	RoPE / ALiBi / sin	Posizione discreta + pair features	$\times$
Simmetria target	S_N (permutaz.)	SE(3) (roto-traslaz.)	$\times$
Struttura features	1D: $N \times d$	1D + 2D: $(N{\times}d, N{\times}N{\times}d)$	$\times$
Obiettivo	Cross-entropy	FAPE + ausiliari geom.	$\times$
Causal mask	Sì	No	$\times$
KV-cache (inferenza)	Essenziale	Non applicabile	$\times$

La tabella 2 evidenzia che il substrato computazionale è condiviso al $\sim 60\%$ delle componenti architetturali. Le differenze non riguardano mai i mattoni fondamentali (attenzione,

normalizzazione, ottimizzazione), ma sempre il *come* questi vengono adattati ai vincoli del dominio fisico.

1.8 §454 Esempio numerico: calcolo esplicito dell'attenzione scalata

Per fissare l'intuizione, consideriamo un esempio con $N = 3$ token, $d_k = 2$ e $H = 1$. Sia la matrice di input (già proiettata):

$$Q = K = V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 2}. \tag{29}$$

Step 1: Matrice dei punteggi.

$$S = \frac{QK^\top}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}. \tag{30}$$

Numericamente $1/\sqrt{2} \approx 0.707$:

$$S \approx \begin{pmatrix} 0.707 & 0 & 0.707 \\ 0 & 0.707 & 0.707 \\ 0.707 & 0.707 & 1.414 \end{pmatrix}.$$

Step 2: Softmax per righe. Per la riga 3 ($q_3 = (1, 1)$):

$$\tilde{a}_{3j} = \text{softmax}(0.707, 0.707, 1.414) = \frac{(e^{0.707}, e^{0.707}, e^{1.414})}{e^{0.707} + e^{0.707} + e^{1.414}} = \frac{(2.028, 2.028, 4.113)}{8.169} \approx (0.248, 0.248, 0.503).$$

Step 3: Output pesato.

$$\text{out}_3 = 0.248 \cdot (1, 0) + 0.248 \cdot (0, 1) + 0.503 \cdot (1, 1) = (0.752, 0.752).$$

Il terzo token, avendo un punteggio $q_3 = (1, 1)$ che si allinea bene con se stesso, riceve un peso di attenzione maggiore verso se stesso (0.503) rispetto agli altri due token (0.248 ciascuno). Questo è il meccanismo fondamentale di *self-attention*: ogni token recupera una media pesata dei valori dell'intero contesto, con pesi proporzionali alla similarità tra query e key.

Osservazione 1.6 (Interpretazione come soft-retrieval)**.** L'attenzione può essere interpretata come un meccanismo di *ricerca soft* in una memoria associativa. Le key K sono le "etichette" degli elementi in memoria, le value V sono i loro "contenuti", e la query Q è l'indice di ricerca. La differenza rispetto a una memoria hard (come una hash table) è che si recupera una *combinazione convessa* di tutti i contenuti, con pesi proporzionali alla similarità query-key. Questa proprietà rende l'attenzione differenziabile rispetto ai parametri, permettendo il training con backpropagation.

1.9 §455a Tokenizzazione: BPE, amino acidi e nucleotidi

La scelta dell'unità atomica di input — il *token* — è una delle decisioni più impattanti sull'architettura, ed è radicalmente diversa nei tre sistemi che stiamo confrontando.

1.9.1 Byte-Pair Encoding (BPE) nei LLM

BPE (Sennrich et al., 2016) è un algoritmo di compressione adattiva applicato al testo. Dato un corpus $\mathcal{C}$, l'algoritmo costruisce iterativamente un vocabolario $\mathcal{V}$ di dimensione V:

Algorithm 2 Byte-Pair Encoding

```
Inizializza V con tutti i caratteri del corpus
Rappresenta ogni parola come sequenza di caratteri separati da spazi
repeat
    Conta tutte le coppie di simboli adiacenti nel corpus
    Trova la coppia più frequente (a, b)
    Unisci: ogni occorrenza di a b → ab
    Aggiungi ab a V
until |V| = V
```

Proprietà chiave di BPE.

- Il vocabolario $\mathcal{V}$ cattura le sottoparole più frequenti della lingua, bilanciando efficienza di rappresentazione e copertura out-of-vocabulary.
- Un testo in inglese con $|\mathcal{V}| \approx 50{,}000$ ha un rapporto di compressione di circa 3–4 caratteri per token (GPT-4 usa $\approx 100{,}000$ token).
- *Conseguenza per l'attenzione*: la lunghezza di sequenza N è inversamente proporzionale all'efficacia di BPE. Un testo di 1000 parole diventa ~ 750 token con GPT-4's tokenizer.

1.9.2 Tokenizzazione aminoacidica in AlphaFold

AlphaFold 2 usa l'alfabeto naturale delle proteine: 20 aminoacidi standard + 1 simbolo speciale per residui sconosciuti/mascherati, per un totale di $|\Sigma| = 21$.

$$e_r = \text{Embedding}(r) \in \mathbb{R}^{c_s}, \quad r \in \{A, C, D, \ldots, Y, \text{mask}\}. \tag{31}$$

Differenza critica con BPE. Il vocabolario aminoacidico è *fisso e universale*: ogni simbolo corrisponde a un'unità chimicamente distinta. Non c'è un trade-off compressione/copertura come in BPE perché il "linguaggio" delle proteine ha un alfabeto naturale di cardinalità piccola e ben definita. Un token = un residuo = un'unità chimica.

1.9.3 Tokenizzazione nucleotidica in AlphaGenome

Per il DNA, l'alfabeto è ancora più piccolo: $\{A, C, G, T\}$ più simboli ambigui IUPAC (N, R, Y, ...). Il problema è la lunghezza: 1 Mbp di DNA = 1 milione di token a livello di nucleotide.

AlphaGenome usa una strategia *ibrida*:

1. A livello di nucleotide singolo (risoluzione 1 bp): one-hot $e_b \in \mathbb{R}^4$
2. Dopo il conv-stem (5 strati, fattore cumulativo 128×): token da 128 bp
3. Per le scale più coarse: token da 1 kbp e 8 kbp

Tabella 3: Confronto dei sistemi di tokenizzazione

Sistema	Vocabolario	Token ≈	Adattivo?	Fisso?	Multi-scala?
GPT-4 (BPE)	∼100k	3–4 caratteri	Sì	No	No
AlphaFold	21	1 residuo	No	Sì	No
AlphaGenome	4 (+ IUPAC)	1 bp → 128 bp	No	Sì	Sì

Implicazione fondamentale della tokenizzazione

La scelta del token è la prima *inductive bias*: definisce l'unità minima di informazione che il modello può rappresentare e manipolare. Nei LLM il token è adattivo e cattura la struttura statistica del testo. In AlphaFold il token coincide con l'unità biochimica naturale. In AlphaGenome il token è gerarchico, riflettendo la struttura multi-scala del genoma. Ogni scelta è matematicamente *necessaria* dato il dominio.

1.10 §455b Analisi del flusso del gradiente negli strati Transformer

La capacità di addestrare reti con molti strati dipende criticamente dalla propagazione del gradiente attraverso le residual connection. Analizziamo quantitativamente questo flusso.

1.10.1 Teorema di propagazione del gradiente con Pre-LN

Consideriamo L blocchi Pre-LN in sequenza. Il gradiente rispetto all'input del blocco l è:

$$\frac{\partial \mathcal{L}}{\partial h^{(l)}} = \frac{\partial \mathcal{L}}{\partial h^{(L)}} \cdot \prod_{k=l}^{L-1} \left(I + J_k\right), \tag{32}$$

dove $J_k = \partial \text{Block}_k / \partial h^{(k)}$ è la Jacobiana del k-esimo blocco (dopo la layer norm).

Teorema 1.7 (Scala del gradiente con Pre-LN)**.** *Con Pre-LN e inizializzazione standard,* $\|J_k\|_F = O(1)$ *indipendentemente dalla profondità* l *e dalla dimensione* d*. Di conseguenza:*

$$\left\| \frac{\partial \mathcal{L}}{\partial h^{(l)}} \right\|_F = \left\| \frac{\partial \mathcal{L}}{\partial h^{(L)}} \right\|_F \cdot \prod_{k=l}^{L-1} (1 + O(\|J_k\|)) = O(1)^{L-l} \cdot \left\| \frac{\partial \mathcal{L}}{\partial h^{(L)}} \right\|_F .$$

Il prodotto si mantiene $O(1)$ perché i termini J_k sono centrati attorno a zero in media, non esplodono né collassano. Questo è matematicamente equivalente alla condizione di *edge of chaos* nella teoria dei sistemi dinamici: la rete opera al margine tra propagazione esplosiva e collasso del gradiente.

1.10.2 Warm-up del learning rate: motivazione matematica

Il warm-up lineare (da 0 a $\alpha_{\max}$ in T_{warm} passi) è motivato dall'analisi del bias delle stime dei momenti in Adam. Al passo $t = 1$:

$$\hat{v}_1 = \frac{(1-\beta_2) g_1^2}{1-\beta_2^1} = g_1^2, \tag{33}$$

che senza warm-up produrrebbe un passo enorme se $\hat{m}_1 \approx g_1$ e $\hat{v}_1 = g_1^2$: $\Delta\theta_1 = \alpha g_1/|g_1| = \alpha \cdot \text{sign}(g_1)$, indipendentemente dalla magnitudine del gradiente. Con il warm-up, il passo al passo t è $\alpha_t = \alpha_{\max} \cdot t / T_{\text{warm}}$, ammortizzando questo effetto mentre le statistiche del momento si stabilizzano.

1.11 §455c Mixture of Experts: routing e bilanciamento del carico

I modelli *Mixture of Experts* (MoE) sostituiscono il singolo blocco FFN con E reti "esperte" $\{F_e\}_{e=1}^{E}$ e un router che seleziona, per ogni token, un sottoinsieme sparse di k esperti.

Tabella 4: Learning rate schedules tipici per LLM e AI scientifica

Schedule	Formula	Usato in	Motivazione
Linear warmup + cosine	$\alpha_t = \alpha_{\max} \cdot f_{\cos}(t)$	LLaMA, GPT	Smooth decay
Linear warmup + exp	$\alpha_t = \alpha_0 e^{-\lambda t}$	AlphaFold 2	Convergenza stabile
Constant + linear decay	$\alpha_t = \alpha_0(1 - t/T)$	AlphaGenome	Semplice, efficace
One-cycle	Peak a 30% del training	Visione	Regularizzazione

1.11.1 Architettura formale

Sia $h \in \mathbb{R}^d$ la rappresentazione di un token. Il router calcola i *logit di gating*:

$$g(h) = \text{softmax}(hW_g) \in \mathbb{R}^E, \qquad W_g \in \mathbb{R}^{d \times E}. \tag{34}$$

Il top-k routing seleziona i k esperti con score più alto:

$$\mathcal{S}(h) = \text{TopK}(g(h), k), \quad \tilde{g}_e(h) = \frac{g_e(h)}{\sum_{e' \in \mathcal{S}(h)} g_{e'}(h)} \cdot \mathbf{1}[e \in \mathcal{S}(h)]. \tag{35}$$

L'output è la combinazione convessa degli esperti selezionati:

$$\text{MoE}(h) = \sum_{e \in \mathcal{S}(h)} \tilde{g}_e(h) \cdot F_e(h). \tag{36}$$

Costo computazionale. Con E esperti e top-k routing, si attivano k/E parametri per token. Il costo per token è $O(k \cdot d_{\text{FFN}} \cdot d)$ invece di $O(E \cdot d_{\text{FFN}} \cdot d)$ per il MoE denso. Modelli come Mixtral-8x7B usano $E = 8$, $k = 2$: ogni token attiva solo il 25% dei parametri FFN.[1]

1.11.2 Il problema del load balancing

Senza regolarizzazione, il router collassa su un piccolo sottoinsieme di esperti (fenomeno di *expert collapse*). La *auxiliary load-balancing loss* (Fedus et al., 2022) penalizza la disuniformità del carico:

Definizione 1.8 (Auxiliary loss per load balancing)**.** Per un batch di T token, sia $f_e = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[e \in \mathcal{S}(h_t)]$ la frequenza con cui l'esperto e è selezionato, e $P_e = \frac{1}{T}\sum_{t=1}^{T} g_e(h_t)$ il suo score medio. La loss è:

$$\mathcal{L}_{\text{aux}} = \alpha E \sum_{e=1}^{E} f_e \cdot P_e, \tag{37}$$

dove $\alpha \approx 10^{-2}$ è un iperparametro.

Proposizione 1.9 (Punto fisso del bilanciamento)**.** *Il minimo di $\mathcal{L}_{aux}$ è raggiunto quando $f_e = 1/E$ per ogni e, cioè quando ogni esperto riceve esattamente T/E token. La loss è quindi una penalità sulla correlazione tra f_e e P_e: se un esperto è frequentemente selezionato (f_e alto), la sua probabilità media P_e deve restare bassa, e viceversa, spingendo verso la distribuzione uniforme.*

[1] Il *parametro totale* è E volte maggiore rispetto a un modello denso di pari costo computazionale, abilitando la cosiddetta *parameter efficiency*: si ottiene la capacità di un modello grande al costo inferenziale di uno piccolo.

1.11.3 MoE in AI scientifica

AlphaFold 2 e AlphaGenome *non* usano MoE. La ragione è strutturale: la lunghezza delle sequenze proteiche ($L \leq 2500$) non rende il risparmio computazionale di MoE critico, e la rappresentazione di coppia z_{ij} ha già una struttura sparse implicita (le coppie fisicamente distanti contribuiscono poco). Per il genoma, AlphaGenome usa una gerarchia di risoluzioni che assolve un ruolo simile: diversi "esperti di scala" sono attivati in base alla risoluzione genomica rilevante.

1.12 §455d Speculative Decoding: inferenza accelerata con drafting

La generazione autoregressiva di un LLM è intrinsecamente sequenziale: il token x_{t+1} dipende da $(x_1, \ldots, x_t)$ e non può essere generato in parallelo. Lo *speculative decoding* (Leviathan et al., 2023; Chen et al., 2023) accelera l'inferenza usando un modello "draft" piccolo q_θ per proporre γ token candidati, poi un modello "verifier" grande p_ϕ per accettarli o rifiutarli in parallelo.

1.12.1 Algoritmo e correttezza

Algorithm 3 Speculative Decoding

Require: Modello target p_ϕ, modello draft q_θ, lunghezza speculazione γ

while non generato EOS **do**
  Draft: genera γ token $(\tilde{x}_1, \ldots, \tilde{x}_\gamma)$ da q_θ
  Verifica: calcola $p_\phi(\tilde{x}_j \mid x_{<t}, \tilde{x}_{<j})$ per $j = 1, \ldots, \gamma$ in **parallelo**
  for $j = 1$ to γ **do**
    $\alpha_j \leftarrow \min\left(1, \frac{p_\phi(\tilde{x}_j)}{q_\theta(\tilde{x}_j)}\right)$
    Accetta $\tilde{x}_j$ con probabilità α_j
    if rifiutato **then**
      Campiona x_{t+j} da $p'_j = \text{normalize}(\max(0, p_\phi - q_\theta))$
      break
    end if
  end for
end while

Teorema 1.10 (Correttezza del Speculative Decoding)**.** *La distribuzione marginale dei token accettati è identica alla distribuzione del modello target p_ϕ.*

Dimostrazione. Sia $\alpha = \min(1, p_\phi/q_\theta)$. La probabilità che un token x venga accettato o campionato dalla distribuzione corretta è:

$$\Pr[\text{output} = x] = q_\theta(x) \cdot \frac{p_\phi(x)}{q_\theta(x)} + \Pr[\text{rejected}] \cdot \frac{\max(0, p_\phi(x) - q_\theta(x))}{\sum_{x'} \max(0, p_\phi(x') - q_\theta(x'))} \tag{38}$$

$$= p_\phi(x). \quad \square \tag{39}$$

□

Accelerazione attesa. Il numero medio di token accettati prima di un rifiuto è:

$$\mathbb{E}[\text{token/passo}] = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}, \quad \alpha = \mathbb{E}_{x \sim q_\theta}\left[\min\left(1, \frac{p_\phi(x)}{q_\theta(x)}\right)\right]. \tag{40}$$

Per $q_\theta \approx p_\phi$ (distribuzione simile), $\alpha \to 1$ e si accettano quasi tutti i token: un'accelerazione di $\gamma\times$ rispetto alla generazione sequenziale pura. Nella pratica, con $\gamma = 4$–8 e un draft model 10–20× più piccolo del target, si ottengono accelerazioni di 2–3×.

Nota. Lo speculative decoding non è applicabile ad AlphaFold o AlphaGenome: entrambi generano output in un singolo forward pass (non autoregressivo). È un'ottimizzazione specifica per la generazione sequenziale, un'altra differenza fondamentale tra LLM e AI scientifica.

1.13 §455e RMSNorm: derivazione e confronto con LayerNorm

La *Root Mean Square Layer Normalization* (RMSNorm, Zhang e Sennrich, 2019) semplifica la LayerNorm (11) eliminando la sottrazione della media:

$$\mathrm{RMSNorm}(x) = \frac{x}{\mathrm{RMS}(x)} \odot \gamma, \qquad \mathrm{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \varepsilon}. \tag{41}$$

Proposizione 1.11 (RMSNorm come proiezione su sfera)**.** *RMSNorm (41) (con $\gamma = \mathbf{1}$) equivale alla proiezione di x sulla sfera $\mathcal{S}^{d-1}(\sqrt{d})$ di raggio $\sqrt{d}$, seguita da una riscalatura per $\sqrt{d}$.*

Confronto con LayerNorm. LayerNorm (11) calcola media ($O(d)$), varianza ($O(d)$), e normalizza ($O(d)$): $3d$ operazioni. RMSNorm calcola solo RMS ($O(d)$) e normalizza ($O(d)$): $2d$ operazioni — una riduzione del 33%. In pratica, la differenza computazionale è trascurabile per un singolo layer, ma con $L = 32$ strati, $d = 4096$, e milioni di token, il risparmio diventa rilevante durante il training.

RMSNorm nell'AI scientifica

AlphaFold 2 usa LayerNorm standard. AlphaGenome adotta Pre-LN con LayerNorm, non RMSNorm. La motivazione è che la struttura delle feature di coppia z_{ij} può avere distribuzioni asimmetriche (non centrate in zero) per cui la sottrazione della media in LayerNorm è effettivamente utile — a differenza degli embedding linguistici che tendono già ad essere centrati.

2 Sezione B — Simmetria, Gruppi e Equivarianza

2.1 §456 Perché la simmetria è fondamentale: teoria del gruppo di simmetria

Qualsiasi modello che apprenda una funzione $f : \mathcal{X} \to \mathcal{Y}$ deve decidere quali trasformazioni dell'input lasciare invariate (o equivarianti) nell'output. Questa non è una scelta stilistica: è una *scelta epistemica* che determina:

(i) quanti parametri servono per rappresentare f (un modello che non sfrutta la simmetria deve imparare la stessa funzione su ogni orbita separatamente);

(ii) la velocità di convergenza (il gradiente è ortogonale alle orbite, riducendo la varianza);

(iii) la generalizzazione fuori distribuzione (le simmetrie fisiche valgono per definizione su tutto il dominio).

Definizione 2.1 (Equivarianza e invarianza)**.** Sia G un gruppo che agisce su $\mathcal{X}$ tramite $\rho_{\mathcal{X}} : G \to \mathrm{Aut}(\mathcal{X})$ e su $\mathcal{Y}$ tramite $\rho_{\mathcal{Y}}$. Una funzione $f : \mathcal{X} \to \mathcal{Y}$ è:

- *G-equivariante* se $f(\rho_{\mathcal{X}}(g)\, x) = \rho_{\mathcal{Y}}(g)\, f(x)$ per ogni $g \in G$, $x \in \mathcal{X}$;
- *G-invariante* se $\rho_{\mathcal{Y}}(g) = \mathrm{id}$, ossia $f(\rho_{\mathcal{X}}(g)\, x) = f(x)$.

Il teorema di Weyl e la costruzione sistematica. Il teorema di Weyl sull'invariante fornisce la ricetta generale: ogni funzione G-invariante su $\mathcal{X}$ può essere scritta come funzione arbitraria degli *invarianti di Reynolds*, dove l'operatore di Reynolds $\mathcal{R}[f](x) = \frac{1}{|G|}\sum_{g\in G} f(g\cdot x)$ (per G finito) produce la media sull'orbita.

Per G continuo (come SE(3)) l'operatore di Reynolds è $\mathcal{R}[f](x) = \int_G f(g\cdot x)\,\mathrm{d}\mu(g)$, dove μ è la misura di Haar. La costruzione di reti equivarianti a SE(3) (IPA, §476) è un'istanza concreta di questo teorema.

2.2 §457 Il gruppo S_N e l'attenzione linguistica

Per un LLM con sequenza di N token, il gruppo di simmetria naturale è S_N, il gruppo delle permutazioni di N elementi, di ordine $N!$.

Teorema 2.2 (S_N-equivarianza dell'attenzione)**.** *L'attenzione scalata senza position encoding (equazione (2)) è S_N-equivariante:*

$$\mathrm{Attn}(\pi X) = \pi\,\mathrm{Attn}(X) \quad \forall \pi \in S_N,$$

dove πX denota la matrice ottenuta permutando le righe di X secondo π.

Dimostrazione. Sia P_π la matrice di permutazione associata a π. Allora: $\pi Q = P_\pi X W^Q$, $\pi K = P_\pi X W^K$, $\pi V = P_\pi X W^V$. La matrice dei punteggi diventa:

$$\frac{(P_\pi Q)(P_\pi K)^\mathsf{T}}{\sqrt{d_k}} = P_\pi \frac{QK^\mathsf{T}}{\sqrt{d_k}} P_\pi^\mathsf{T}.$$

Il softmax per righe permuta righe e colonne: $\mathrm{softmax}(P_\pi S P_\pi^\mathsf{T}) = P_\pi\,\mathrm{softmax}(S)P_\pi^\mathsf{T}$. Quindi $\mathrm{Attn}(\pi X) = P_\pi\,\mathrm{softmax}(S)P_\pi^\mathsf{T} P_\pi V = P_\pi A V = \pi\,\mathrm{Attn}(X)$. □ □

Osservazione 2.3. Il position encoding rompe questa equivarianza *intenzionalmente*: aggiungere $\mathrm{PE}(p)$ al token in posizione p rende il modello consapevole dell'ordine nella sequenza, trasformando la simmetria S_N in una simmetria banale (solo l'identità).

Struttura di S_N come gruppo finito. S_N è un gruppo finito non abeliano per $N \geq 3$, generato da trasposizioni adiacenti $(i, i+1)$. L'ordine è $N! = \prod_{k=1}^{N} k$. Le rappresentazioni irriducibili di S_N sono indicizzate da partizioni di N (diagrammi di Young) tramite il teorema di Frobenius. In pratica, per la MHA si sfrutta solo l'equivarianza alla permutazione (la rappresentazione standard) e non le rappresentazioni di ordine superiore.

2.3 §458 Il gruppo SE(3) e la struttura proteica

Una struttura proteica è definita da un insieme di coordinate atomiche $\{r_i\}_{i=1}^{N_{\text{atoms}}} \subset \mathbb{R}^3$. La funzione chiave che AlphaFold deve apprendere è:

$$f : (\text{sequenza amminoacidica} + \text{MSA}) \longrightarrow \{r_i\} \subset \mathbb{R}^3. \tag{42}$$

Il gruppo speciale euclideo. Il gruppo delle roto-traslazioni in $\mathbb{R}^3$ è il gruppo speciale euclideo:

$$\mathrm{SE}(3) = \mathrm{SO}(3) \ltimes \mathbb{R}^3 = \{(R, t) : R \in \mathrm{SO}(3), t \in \mathbb{R}^3\}, \tag{43}$$

dove $\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : R^\mathsf{T}R = I, \det R = 1\}$ è il gruppo delle rotazioni proprie. Il simbolo $\ltimes$ denota il prodotto semidiretto: la legge di composizione è $(R_1, t_1) \circ (R_2, t_2) = (R_1R_2, R_1t_2 + t_1)$. L'azione di $(R, t) \in \mathrm{SE}(3)$ sulle coordinate è:

$$(R, t) \cdot r_i = Rr_i + t. \tag{44}$$

Definizione 2.4 (SE(3) come gruppo di Lie)**.** SE(3) è un gruppo di Lie non compatto di dimensione 6 (3 rotazioni + 3 traslazioni). La sua algebra di Lie $\mathfrak{se}(3)$ ha base $\{J_x, J_y, J_z, P_x, P_y, P_z\}$ con relazioni di commutazione:

$$[J_i, J_j] = \varepsilon_{ijk} J_k, \quad [J_i, P_j] = \varepsilon_{ijk} P_k, \quad [P_i, P_j] = 0,$$

dove ε_{ijk} è il tensore di Levi-Civita.

Rappresentazioni irriducibili di SO(3)**: armoniche sferiche.** Le rappresentazioni irriducibili di SO(3) sono indicizzate dal numero quantico angolare $\ell = 0, 1, 2, \ldots$, con dimensione $2\ell + 1$. I vettori di base sono le armoniche sferiche reali $Y_\ell^m : S^2 \to \mathbb{R}$, $m = -\ell, \ldots, \ell$.

Sotto una rotazione $R \in \mathrm{SO}(3)$:

$$Y_\ell^m(R^{-1}\hat{n}) = \sum_{m'=-\ell}^{\ell} D^\ell_{m'm}(R)\, Y_\ell^{m'}(\hat{n}), \tag{45}$$

dove $D^\ell(R)$ è la matrice di Wigner di ordine ℓ. Le reti equivarianti SE(3) (come e3nn, NequIP) usano features ℓ-vettoriali che si trasformano secondo (45).

2.3.1 Perché AlphaFold NON usa armoniche sferiche

Le reti SE(3)-equivarianti basate su armoniche sferiche (e3nn, NequIP, SEGNN) sono *esattamente* equivarianti ma hanno costo computazionale $O(L^3)$ per il tensore di Clebsch-Gordan. AlphaFold 2 usa invece un approccio alternativo: imporre l'invarianza tramite *frames locali* (§460) e l'*Invariant Point Attention* (§476), che costruisce features già invarianti senza fare uso esplicito di armoniche. Il costo è $O(L^2)$ nell'attenzione.

Tabella 5: Confronto tra approcci all'equivarianza SE(3)

Approccio	Metodo	Costo	Esempio
Armoniche sferiche	Features ℓ-vettoriali	$O(L^3 N)$	NequIP, SEGNN
Frame + invarianti	Coordinate locali	$O(L^2 N)$	AlphaFold 2 IPA
Data augmentation	Invarianza in media	$O(LN)$	AlphaGenome WC
Invarianti scalari	Distanze e angoli	$O(L^2)$	SchNet

2.4 §459 Confronto algebrico: S_N vs SE(3)

La differenza è strutturale: mentre nei LLM si usa una simmetria discreta che viene esplicitamente rotta aggiungendo informazione posizionale, in AlphaFold si deve preservare l'invarianza *continua* di SE(3) per tutta la pipeline fino alla struttura finale. L'invarianza continua è infinitamente più stringente: non basta soddisfarla su un insieme discreto di trasformazioni, ma su un gruppo di cardinalità continua.

2.5 §460 Frames rigidi e rappresentazione del backbone

AlphaFold 2 introduce la nozione di *frame rigido* per rappresentare l'orientamento locale di ogni residuo.

Definizione 2.5 (Frame rigido)**.** Un frame rigido per il residuo i è una coppia $T_i = (R_i, t_i)$ con $R_i \in \mathrm{SO}(3)$ e $t_i \in \mathbb{R}^3$, dove:

- t_i è la posizione del C^α del residuo i;
- R_i codifica l'orientamento del piano del backbone N-C^α-C.

Tabella 6: Confronto tra i gruppi di simmetria di LLM e AlphaFold

Proprietà	S_N **(LLM)**	SE(3) **(AlphaFold)**
Tipo	Gruppo finito discreto	Gruppo di Lie non compatto
Cardinalità	$N!$ (finito)	∞ (continuo)
Struttura	Non abeliano per $N \geq 3$	Non abeliano
Dimensione Lie algebra	—	6 (3 rot. + 3 trasl.)
Rappresentaz. naturale	Matrici di permutazione $N \times N$	$Rx + t$
Rappresentaz. irriducibili	Indicizzate da partizioni (Young)	Indicizzate da $\ell \in \mathbb{N}$
Come si realizza	PE che la rompe	IPA che la rispetta (§476)
Costo di imposition	$O(N)$ (somma)	$O(L^2)$ (attenzione IPA)

Costruzione del frame dal backbone. Dati i tre atomi del backbone del residuo i: N_i, C_i^α, C_i, il frame si costruisce come segue:

$$e_1 = \frac{C_i^\alpha - N_i}{\|C_i^\alpha - N_i\|}, \tag{46}$$

$$e_1' = \frac{C_i - N_i}{\|C_i - N_i\|}, \tag{47}$$

$$e_3 = \frac{e_1 \times e_1'}{\|e_1 \times e_1'\|}, \tag{48}$$

$$e_2 = e_3 \times e_1, \tag{49}$$

$$R_i = [e_1|e_2|e_3], \quad t_i = C_i^\alpha. \tag{50}$$

Coordinate locali. Ogni frame T_i definisce un sistema di riferimento locale. La posizione di qualsiasi atomo a nel sistema di riferimento locale i è:

$$x_a^{(i)} = T_i^{-1} \cdot x_a = R_i^\mathsf{T}(x_a - t_i). \tag{51}$$

Proposizione 2.6 (Invarianza delle coordinate locali)**.** *Se si applica una trasformazione globale* $(R_g, t_g) \in \mathrm{SE}(3)$ *a tutto il sistema (tutti gli atomi e tutti i frames), allora* $x_a^{(i)}$ *rimane invariato:*

$$T_i' = (R_g R_i, R_g t_i + t_g), \quad x_a' = R_g x_a + t_g \implies (T_i')^{-1} \cdot x_a' = R_i^\mathsf{T} R_g^\mathsf{T}(R_g x_a + t_g - R_g t_i - t_g) = R_i^\mathsf{T}(x_a - t_i).$$

Questa proposizione è la chiave dell'SE(3)-equivarianza di AlphaFold: *tutte le quantità critiche vengono calcolate in coordinate locali che sono invarianti rispetto a trasformazioni globali del sistema di riferimento.*

Nota. Nei LLM non esiste un concetto analogo ai frames rigidi: ogni token ha un embedding vettoriale in $\mathbb{R}^d$, senza alcuna struttura geometrica interna. Il testo scritto non ha un orientamento fisico, mentre ogni residuo amminoacidico ha un orientamento ben definito nel piano del backbone — una proprietà fondamentale della chimica organica (legame peptidico planare).

2.6 §461–464 Costruire funzioni SE(3)-invarianti: dal primo principio

Per costruire features invarianti a SE(3) si sfruttano *invarianti geometrici*: quantità che non cambiano sotto roto-traslazioni.

2.6.1 Invarianti scalari di primo ordine

Dati due punti $r_i, r_j \in \mathbb{R}^3$, l'unico invariante scalare è la distanza:

$$d_{ij} = \|r_i - r_j\|. \tag{52}$$

Dati tre punti, gli invarianti scalari aggiuntivi sono gli angoli:

$$\cos\theta_{ijk} = \frac{(r_j - r_i)\cdot(r_k - r_i)}{\|r_j - r_i\|\|r_k - r_i\|}. \tag{53}$$

Dati quattro punti, anche gli angoli diedrali φ_{ijkl} (angoli tra due piani definiti da triplette di punti) sono invarianti scalari, ed è tramite questi che AlphaFold codifica la struttura del backbone.

2.6.2 Teorema di completezza degli invarianti

Teorema 2.7 (Completezza degli invarianti per nuvole di punti)**.** *Due nuvole di punti* $\{r_i\}_{i=1}^N$ *e* $\{r'_j\}_{j=1}^N$ *sono related da una trasformazione* SE(3) *se e solo se le loro matrici di distanze interatomiche* $D_{ij} = \|r_i - r_j\|$ *sono identiche.*

Sketch. ($\Rightarrow$) Le distanze sono invarianti per definizione di SE(3). ($\Leftarrow$) Si usa l'embedding multidimensionale (MDS): D determina univocamente la nuvola di punti a meno di SE(3) tramite la fattorizzazione della matrice di Gram $G_{ij} = r_i \cdot r_j$. □ □

2.6.3 Costruzione delle features di coppia iniziali in AlphaFold

AlphaFold usa distanze e angoli diedrali φ, ψ, ω come features di input, e costruisce le features di coppia $z_{ij} \in \mathbb{R}^{c_z}$ inizialmente come:

$$z_{ij}^{(0)} = \text{Linear}(a_i) + \text{Linear}(a_j) + \text{relpos}(i-j), \tag{54}$$

dove $a_i \in \mathbb{R}^{c_s}$ è l'embedding del residuo i e $\text{relpos}(i-j)$ è l'embedding della posizione relativa nella *sequenza* (non nello spazio 3D). Nella fase iniziale, prima dell'Evoformer, non si ha ancora accesso alle coordinate 3D — queste verranno prodotte dallo Structure Module.

2.6.4 Il numero minimo di invarianti necessari

Proposizione 2.8. *Per una catena di L residui proteici, un set completo di invarianti che determina univocamente la struttura (a meno di* SE(3)*) consiste di:*

- $L-1$ *distanze* C_i^α-C_{i+1}^α *(quasi costanti,* ≈ 3.8 *Å);*
- $L-2$ *angoli* $\theta_{i,i+1,i+2}$*;*
- $L-3$ *angoli diedrali* φ_i *(backbone);*
- $L-3$ *angoli diedrali* ψ_i *(backbone);*
- *Angoli* χ *delle catene laterali (fino a 4 per residuo).*

Totale: $O(L)$ *scalari. La rappresentazione di coppia* z_{ij} *con* $O(L^2)$ *voci è ridondante per la struttura, ma cattura correlazioni di lungo raggio in modo computazionalmente efficiente.*

2.6.5 La biforcazione tra S_N e SE(3) come teorema

Teorema 2.9 (Necessità della geometria fisica)**.** *Sia $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$ la famiglia di tutte le reti Transformer. Le seguenti affermazioni sono necessariamente vere:*

(a) Se $\mathcal{Y} = \{1, \ldots, V\}^T$ (sequenze di token), allora il gruppo di simmetria naturale è S_N e ogni tentativo di imporre SE(3)*-invarianza è non solo superfluo ma* impossibile *(non c'è azione di* SE(3) *su sequenze discrete).*

(b) Se $\mathcal{Y} = \mathbb{R}^{3N}$ (coordinate atomiche) con perdita SE(3)*-invariante, allora qualsiasi apprendimento efficiente richiede l'utilizzo di invarianti geometrici che codifichino la struttura metrica di $\mathbb{R}^3$.*

Questo teorema formalizza la tesi centrale del volume: le differenze architetturali tra LLM e AlphaFold non sono scelte di design ma *teoremi* derivati dalla geometria dell'output.

2.7 §462 Equivariant Graph Neural Networks: costruzione esplicita

Le reti EGNN (Satorras et al., 2021) costruiscono l'equivarianza a $E(n)$ (il gruppo di isometrie di $\mathbb{R}^n$ che include rotazioni, riflessioni e traslazioni) in modo diretto tramite un vincolo sull'aggiornamento delle coordinate.

2.7.1 Messaggio equivariante

Sia $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un grafo con n nodi. A ogni nodo i sono associati:

- una feature scalare (invariante) $h_i \in \mathbb{R}^{d_h}$;
- una coordinata vettoriale (equivariante) $x_i \in \mathbb{R}^3$.

L'aggiornamento equivariante dell'EGNN è:

$$m_{ij} = \phi_e\big(h_i, h_j,\ \|x_i - x_j\|^2,\ a_{ij}\big)\,, \tag{55}$$

$$h_i' = \phi_h\left(h_i,\ \sum_{j \in \mathcal{N}(i)} m_{ij}\right), \tag{56}$$

$$x_i' = x_i + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (x_i - x_j)\, \phi_x(m_{ij}), \tag{57}$$

dove ϕ_e, ϕ_h, ϕ_x sono reti neurali scalari e a_{ij} sono attributi scalari dell'arco.

Teorema 2.10 ($E(n)$-equivarianza dell'EGNN)**.** *L'aggiornamento* (57) *è $E(n)$-equivariante, cioè:*

$$\mathrm{EGNN}(Rx + t) = R\,\mathrm{EGNN}(x) + t \quad \forall R \in O(n), t \in \mathbb{R}^n.$$

Dimostrazione. Sotto (R, t): $x_i \mapsto Rx_i + t$, $x_j \mapsto Rx_j + t$.
Invarianza del messaggio. $\|x_i' - x_j'\|^2 = \|(Rx_i + t) - (Rx_j + t)\|^2 = \|R(x_i - x_j)\|^2 = \|x_i - x_j\|^2$, poiché R è un'isometria. Quindi m_{ij} è invariante a (R, t).
Equivarianza dell'aggiornamento di x.

$$(x_i' + t)_{\text{aggiornata}} = Rx_i + t + \frac{1}{|\mathcal{N}|} \sum_j \underbrace{(Rx_i + t - Rx_j - t)}_{R(x_i - x_j)} \phi_x(m_{ij}) = R\left(x_i + \frac{1}{|\mathcal{N}|} \sum_j (x_i - x_j)\phi_x(m_{ij})\right) + t. \quad \square$$

$\square$

Costo computazionale. Ogni messaggio (55) usa solo la distanza $\|x_i - x_j\|^2$ (uno scalare), non il vettore $x_i - x_j$. Di conseguenza la complessità è $O(|\mathcal{E}| \cdot d_h^2)$, lineare nel numero di archi e senza il fattore cubico delle armoniche sferiche.

2.7.2 Confronto con le architetture basate su armoniche sferiche

Le reti SE(3)-equivarianti basate su armoniche sferiche (NequIP, SEGNN) usano feature ℓ-vettoriali $f^{(\ell)} \in \mathbb{R}^{2\ell+1}$ che si trasformano secondo la rappresentazione di Wigner (45). La combinazione di due feature è il prodotto tensoriale di Clebsch-Gordan:

$$(f^{(\ell_1)} \otimes_{CG} f^{(\ell_2)})^{(\ell_3)}_m = \sum_{m_1, m_2} C^{\ell_3 m}_{\ell_1 m_1, \ell_2 m_2}\, f^{(\ell_1)}_{m_1} f^{(\ell_2)}_{m_2}, \tag{58}$$

dove $C^{\ell_3 m}_{\ell_1 m_1, \ell_2 m_2}$ sono i coefficienti di Clebsch-Gordan. Il costo di ogni operazione (58) è $O(L_{\max}^3)$, dove $L_{\max}$ è l'ordine massimo delle armoniche. Per $L_{\max} = 3$ (comune in NequIP), il costo per arco è $O(27 \cdot d_h)$, confrontato con $O(d_h^2)$ per EGNN.

2.8 §463 Dalla distanza alle feature equivarianti: SchNet, DimeNet, PaiNN

Prima delle reti equivarianti complete, una famiglia di modelli ha sfruttato invarianti scalari per modellare sistemi molecolari. Li esaminiamo come base di confronto.

2.8.1 SchNet (Schütt et al., 2017)

SchNet usa la distanza interatomica $d_{ij} = \|r_i - r_j\|$ come unica feature geometrica. L'aggiornamento del messaggio è:

$$m_{ij} = h_j \odot W_\phi(\mathrm{RBF}(d_{ij})), \qquad \mathrm{RBF}(d)_k = e^{-\gamma(d-\mu_k)^2}, \tag{59}$$

dove le funzioni radiali di base (RBF) con centri $\{\mu_k\}$ discretizzano la distanza. SchNet è $O(3)$-invariante ma *non* equivariante: produce solo predizioni scalari (energia, carica, ecc.), non vettori o tensori.

2.8.2 DimeNet (Klicpera et al., 2020)

DimeNet estende SchNet includendo gli angoli di legame tramite funzioni di base sferiche (senza spinori completi):

$$m_{ij} = \phi_{\mathrm{msg}}(\mathrm{RBF}(d_{ij}),\ \mathrm{SBF}(d_{ji}, \theta_{ijk})), \tag{60}$$

dove $\mathrm{SBF}(d, \theta) = e_n(d) \cdot Y_l^0(\theta)$ combina informazione radiale e angolare (ma non ordine $|m| > 0$, quindi non completo rotationally).

2.8.3 PaiNN (Schütt et al., 2021)

PaiNN introduce feature vettoriali $\vec{v}_i \in \mathbb{R}^{3\times d}$ accanto a quelle scalari $s_i \in \mathbb{R}^d$, costruendo l'equivarianza tramite l'operatore differenziale $\hat{r}_{ij} = (r_i - r_j)/\|r_i - r_j\|$:

$$\Delta s_i = \sum_j \phi_s(s_j, \|r_{ij}\|) \odot W_s(r_{ij}), \tag{61}$$

$$\Delta \vec{v}_i = \sum_j \phi_v(s_j, \|r_{ij}\|) \otimes \hat{r}_{ij}, \tag{62}$$

ottenendo una rete $O(3)$-equivariante per vettori di primo ordine. PaiNN è più espressivo di SchNet e DimeNet ma meno completo di NequIP (che usa ordini arbitrari).

Tabella 7: Confronto delle architetture equivarianti molecolari

Modello	**Feature**	**Equivarianza**	**Costo**	**Anno**
SchNet	Solo scalari	$O(3)$-invariante	$O(\|\mathcal{E}\|)$	2017
DimeNet	Scalari + angoli	Parziale	$O(\|\mathcal{E}\| \cdot d^2)$	2020
PaiNN	Scalari + vettori	$O(3)$-equivar. ($\ell \leq 1$)	$O(\|\mathcal{E}\| \cdot d)$	2021
EGNN	Scalari + coord.	$E(n)$-equivar.	$O(\|\mathcal{E}\| \cdot d^2)$	2021
NequIP	Armoniche ($\ell \leq L$)	$O(3)$-equivar. (completa)	$O(\|\mathcal{E}\| \cdot L^3 d)$	2022
AlphaFold IPA	Frame rigidi	SE(3)-invar.	$O(L^2 d)$	2021

2.9 §464 Come l'IPA di AlphaFold si inserisce nel framework equivariante

L'Invariant Point Attention (IPA, §476) è formalmente un'istanza del meccanismo di attenzione su grafi con feature invarianti a SE(3). La differenza rispetto alle reti equivarianti descritte sopra è nell'*output*:

- Le reti equivarianti (NequIP, EGNN) producono feature vettoriali SE(3)-equivarianti come output intermedi;
- L'IPA produce feature scalari SE(3)-*invarianti* come output del meccanismo di attenzione, e poi aggiorna i frames (equivarianti) separatamente nel modulo di backbone update.

Questa scelta divide rigorosamente il problema in due sottoproblemi:

1. **Aggiornamento delle features scalari**: risolto da IPA in modo invariante — le features s_i non hanno "orientamento", sono scalari.
2. **Aggiornamento dei frames**: risolto dal backbone update tramite quaternioni (§472-475) — i frames T_i si aggiornano in modo equivariante.

Proposizione 2.11 (Equivarianza del ciclo completo Evoformer + Structure Module)**.** *Sia* $(s, z) \mapsto \{T_i, x_a\}$ *la pipeline completa di AlphaFold 2. Questa funzione è* SE(3)*-equivariante: se le coordinate iniziali* $\{x_a^{(0)}\}$ *vengono trasformate da* (R_g, t_g)*, allora le coordinate predette si trasformano anch'esse da* (R_g, t_g)*.*

Dimostrazione. Il precedente teorema di invarianza dell'IPA garantisce che s_i non cambia sotto SE(3). L'aggiornamento del backbone calcola T_i in modo che dipenda solo dalle feature scalari s_i (invarianti) e dalle coordinate relative nel frame locale (invarianti per la Proposizione §460). Quindi i frames predetti T_i si trasformano coerentemente con una trasformazione globale (R_g, t_g), e le coordinate atomiche ricostruite tramite (86) sono SE(3)-equivarianti. □ □

2.10 §464b Vantaggio in complessità campionaria dei modelli equivarianti

Un argomento frequente a favore dei modelli equivarianti è che "sfruttano la simmetria come dato aggiuntivo gratuito". Formalizziamo questo con un bound di complessità campionaria.

Definizione 2.12 (Classe di ipotesi equivariante)**.** Sia $\mathcal{H}_G$ la classe di tutte le funzioni $f : \mathcal{X} \to \mathcal{Y}$ continue e G-equivarianti, e $\mathcal{H}$ la classe di tutte le funzioni continue senza vincoli di simmetria.

Teorema 2.13 (Riduzione della complessità Rademacher per equivarianza)**.** *Sia* $\mathfrak{R}_m(\mathcal{H})$ *la complessità di Rademacher di* $\mathcal{H}$ *su* m *campioni. Allora:*

$$\mathfrak{R}_m(\mathcal{H}_G) \leq \frac{\mathfrak{R}_m(\mathcal{H})}{\sqrt{|G|}}, \tag{63}$$

dove $|G|$ *è l'ordine del gruppo* G *(per* G *finito).*

Sketch. Ogni $f \in \mathcal{H}_G$ può essere decomposta come $f(x) = \frac{1}{|G|} \sum_{g \in G} \tilde{f}(g \cdot x)$ per qualche $\tilde{f} \in \mathcal{H}$ (averaging trick). Per la disuguaglianza di Jensen e le proprietà della complessità di Rademacher: $\mathfrak{R}_m(\mathcal{H}_G) \leq (1/|G|) \cdot |G| \cdot \mathfrak{R}_m(\mathcal{H})/\sqrt{|G|} = \mathfrak{R}_m(\mathcal{H})/\sqrt{|G|}$. □ □

Interpretazione pratica. Per $G = S_N$ con $N = 100$ (100 atomi): $|G| = 100! \approx 10^{158}$. Anche se il bound (63) è quasi certamente non stretto per gruppi così grandi, l'idea qualitativa è corretta: imporre G-equivarianza riduce l'iperspazio di ricerca delle funzioni apprendibili.

Per $G = \mathrm{SE}(3)$ (gruppo continuo), l'argomento si generalizza usando la misura di Haar: il fattore di riduzione è proporzionale alla *misura del gruppo*, che per $\mathrm{SO}(3)$ è $8\pi^2$ (il volume di $\mathrm{SO}(3)$ con la metrica di Haar normalizzata). Questo è il fondamento teorico dell'affermazione che AlphaFold 2, usando IPA e frames rigidi, "vale" come enormi quantità di dati strutturali aggiuntivi — la simmetria è informazione gratuita.

Tabella 8: Vantaggio in complessità campionaria per diversi gruppi

Gruppo	Modello	Fattore di riduzione	Equivalente dati
S_N $(N = 100)$	LLM senza PE	$\sqrt{100!} \approx 10^{79}$	astronomico
$\mathrm{SO}(3)$	AlphaFold IPA	$\sqrt{8\pi^2} \approx 8.9$	$\sim 9\times$ dati in più
$\mathbb{Z}_2$	AlphaGenome WC	$\sqrt{2} \approx 1.4$	$\sim 1.4\times$ dati in più
$O(3)$	EGNN	$\sqrt{16\pi^2/3} \approx 7.3$	$\sim 7\times$ dati in più

2.11 §464c Espressività delle GNN: la gerarchia Weisfeiler-Lehman

Prima dell'IPA e delle architetture equivarianti, le reti molecolari erano formulate come Graph Neural Network (GNN). La gerarchia di Weisfeiler-Lehman (WL) quantifica l'espressività massima di queste reti.

2.11.1 Il test di isomorfismo WL

Il test 1-WL (Color Refinement) assegna iterativamente colori ai nodi di un grafo basandosi sul colore del nodo e di tutti i suoi vicini:

$$c_v^{(t+1)} = \mathrm{HASH}(c_v^{(t)},\ \{\{c_u^{(t)} : u \in \mathcal{N}(v)\}\}), \tag{64}$$

dove $\{\{\cdot\}\}$ è un multiset. Due grafi sono distinti da 1-WL se e solo se hanno sequenze di colori finali diverse.

Teorema 2.14 (GNN = 1-WL (Xu et al., 2019))**.** *Qualsiasi GNN con aggregazione di messaggi di tipo* $h_v^{(t+1)} = \mathrm{UPDATE}(h_v^{(t)},\ \mathrm{AGG}(\{h_u^{(t)} : u \in \mathcal{N}(v)\}))$ *ha potere discriminativo al massimo uguale al test 1-WL. Esiste una GNN (Graph Isomorphism Network, GIN) che lo raggiunge:*

$$h_v^{(t+1)} = \mathrm{MLP}^{(t)}\left((1+\varepsilon)\, h_v^{(t)} + \sum_{u \in \mathcal{N}(v)} h_u^{(t)}\right). \tag{65}$$

Limitazioni per la geometria molecolare. 1-WL non può distinguere due grafi che hanno la stessa topologia ma diverse geometrie 3D (ad esempio, enantiomeri: molecole che sono immagini speculari l'una dell'altra). Questo è il motivo per cui le GNN scalari (SchNet, GIN) non possono predire proprietà chirali o fenomeni 3D.

Le reti equivarianti a SE(3) (NequIP, IPA) corrispondono a test WL di ordine superiore che includono informazione geometrica: il colore del nodo include non solo la sua identità chimica ma anche la sua *posizione relativa* nel sistema di riferimento locale — il principio fondamentale dei frames rigidi.

IPA come GNN geometrica di ordine superiore

L'IPA di AlphaFold è formalmente una GNN con messaggi che includono: (1) features scalari s_i, s_j (come 1-WL); (2) features di coppia z_{ij} (come 2-WL); (3) distanze 3D $\|q^{3D}_{ihk} - k^{3D}_{jhk}\|^2$ (informazione geometrica). I punti (2) e (3) la collocano in una gerarchia WL superiore a quella delle GNN scalari, giustificando la sua superiorità su approcci basati solo su distanze.

3 Sezione C — AlphaFold 2: Architettura Matematica Completa

3.1 §465 Panoramica architetturale e notazione

AlphaFold 2 (Jumper et al., *Nature* 2021) predice la struttura 3D di una proteina dalla sua sequenza amminoacidica. L'architettura si compone di tre stadi:

1. **Feature extraction**: costruzione delle rappresentazioni di sequenza $s_i \in \mathbb{R}^{c_s}$ e di coppia $z_{ij} \in \mathbb{R}^{c_z}$.
2. **Evoformer** ($N_{\text{evo}} = 48$ blocchi): raffinamento iterativo di s e z attraverso attenzione assiale, aggiornamenti triangolari e comunicazione bidirezionale.
3. **Structure Module** ($N_{\text{str}} = 8$ blocchi): predizione iterativa dei frames T_i e degli angoli torsionali.

Notazione. L: lunghezza della sequenza; N_{seq}: numero di sequenze nell'MSA; $c_s = 384$, $c_z = 128$, $c_m = 256$ (dimensioni in AF2); Σ: alfabeto di 21 simboli; $\odot$: prodotto di Hadamard; $\otimes$: prodotto esterno.

La pipeline completa:

$$(s^{(0)}, z^{(0)}) \xrightarrow{\text{Evoformer}} (s^{(N_{\text{evo}})}, z^{(N_{\text{evo}})}) \xrightarrow{\text{Str. Mod.}} \{T_i\}_{i=1}^{L} \xrightarrow{\text{backbone}} \{r_a\}. \tag{66}$$

3.2 §466 Multiple Sequence Alignment come memoria evolutiva

L'MSA è una matrice $M \in \Sigma^{N_{\text{seq}} \times L}$, dove la prima riga è la sequenza target e le righe successive sono sequenze omologhe trovate in database evolutivi (UniRef90, BFD, MGnify).

La coevoluzione come segnale di struttura. Se i residui i e j interagiscono fisicamente nel fold, la loro *co-variazione evolutiva* è un segnale diretto. La matrice di covarianza:

$$C_{ij} = \frac{1}{N_{\text{seq}}} \sum_k (M_{ki} - \bar{M}_i)(M_{kj} - \bar{M}_j) \tag{67}$$

ha picchi alle coppie (i, j) in contatto nella struttura 3D. AlphaFold lascia che la rete apprenda implicitamente queste covariazioni attraverso l'attenzione sull'MSA, anziché usare C direttamente come i metodi classici DCA.

Embedding dell'MSA.

$$m_{si} = \text{Linear}(\text{onehot}(M_{si})) + e_s + e_i, \quad s = 1, \ldots, N_{\text{seq}}, \quad i = 1, \ldots, L. \tag{68}$$

La rappresentazione MSA è un tensore $\mathcal{M} \in \mathbb{R}^{N_{\text{seq}} \times L \times c_m}$.

Nota. Il concetto di MSA non ha equivalente nei LLM. La differenza chiave è la semantica: nell'MSA ogni riga è una variante evolutiva della stessa proteina, con covarianze inter-colonna che codificano le interazioni fisiche fra residui. Nei LLM non esistono "sequenze parallele" della stessa frase.

3.3 §467 Attenzione assiale nell'Evoformer

Applicare attenzione completa su $N_{\text{seq}} \cdot L$ elementi sarebbe $O(N_{\text{seq}}^2 L^2)$. L'Evoformer usa *attenzione assiale*: separatamente lungo le righe (asse sequenza) e lungo le colonne (asse specie).

3.3.1 MSA row-wise attention con pair bias

$$A_{h,ij}^{(\text{row})} = \text{softmax}_j\left(\frac{1}{\sqrt{d_h}}\langle W_h^Q m_{si}, W_h^K m_{sj}\rangle + b_{h,ij}^{(\text{pair})}\right), \tag{69}$$

dove $b_{h,ij}^{(\text{pair})} = \text{Linear}_h(z_{ij}) \in \mathbb{R}$. Il *pair-biased attention* permette alle informazioni strutturali di guidare l'aggiornamento dell'MSA. Complessità: $O(N_{\text{seq}} L^2 d_h)$.

3.3.2 MSA column-wise attention

$$A_{h,ss'}^{(\text{col})} = \text{softmax}_{s'}\left(\frac{1}{\sqrt{d_h}}\langle W_h^Q m_{si}, W_h^K m_{s'i}\rangle\right), \quad \text{(fissato } i\text{)}. \tag{70}$$

Complessità: $O(L N_{\text{seq}}^2 d_h)$.

3.3.3 Outer product mean

$$\Delta z_{ij} = \text{Linear}\left(\frac{1}{N_{\text{seq}}}\sum_s a_{si} \otimes a_{sj}\right), \quad a_{si} = \text{Linear}(m_{si}) \in \mathbb{R}^{c_a}. \tag{71}$$

Il prodotto esterno $(u \otimes v)_{k_1 k_2} = u_{k_1} v_{k_2}$ cattura le correlazioni statistiche fra i residui i e j — la realizzazione differenziabile della co-varianza (67).

Un blocco Evoformer completo. Un singolo blocco Evoformer applica in sequenza: (1) Row-wise MSA attention con pair bias; (2) Column-wise MSA attention; (3) MSA transition (FFN); (4) Outer product mean ($\mathcal{M} \to z$); (5) Triangular multiplicative update outgoing; (6) Triangular multiplicative update incoming; (7) Triangular self-attention starting node; (8) Triangular self-attention ending node; (9) Pair transition (FFN).

3.4 §468–471 Aggiornamenti triangolari: la matematica dell'invariante geometrico

Gli aggiornamenti triangolari sono la struttura più originale di AlphaFold 2 e non hanno equivalente in alcun LLM. Sfruttano la *disuguaglianza triangolare*: $|d_{ik} - d_{jk}| \le d_{ij} \le d_{ik} + d_{jk}$.

3.4.1 Aggiornamento moltiplicativo outgoing

$$a_{ij} = \sigma(\text{Linear}(z_{ij})) \odot \text{Linear}(z_{ij}) \in \mathbb{R}^c, \tag{72}$$

$$b_{ij} = \sigma(\text{Linear}(z_{ij})) \odot \text{Linear}(z_{ij}) \in \mathbb{R}^c, \tag{73}$$

$$\tilde{z}_{ij}^{(\text{out})} = \text{LayerNorm}\left(g_{ij} \odot \text{Linear}\left(\sum_{k=1}^{L} a_{ik} \odot b_{jk}\right)\right), \tag{74}$$

dove $g_{ij} = \sigma(\text{Linear}(z_{ij}))$ è una gate globale. Il termine $\sum_k a_{ik} \odot b_{jk}$ integra le informazioni su tutti gli intermediari k nel percorso $i \to k \to j$.

3.4.2 Aggiornamento moltiplicativo incoming

$$\tilde{z}_{ij}^{(\text{in})} = \text{LayerNorm}\left(g_{ij} \odot \text{Linear}\left(\sum_{k=1}^{L} a_{ki} \odot b_{kj} \right) \right). \tag{75}$$

Considera i percorsi $k \to i \to j$ (con k che entra in i).

3.4.3 Interpretazione geometrica

Proposizione 3.1 (Consistenza metrica degli aggiornamenti triangolari)**.** *Se z_{ij} approssima una funzione della distanza d_{ij}, allora l'aggiornamento outgoing $\sum_k a_{ik} \odot b_{jk}$ implementa una forma di propagazione di vincoli metrici: aggiorna la stima di d_{ij} tenendo conto di tutti i percorsi indiretti $i \to k \to j$, il che tende a far convergere la rappresentazione verso una metrica euclidea consistente.*

3.4.4 Attenzione triangolare (starting node)

$$A_{h,jk}^{(\text{start})} = \text{softmax}_k\left(\frac{1}{\sqrt{d_h}} \langle W_h^Q z_{ij}, W_h^K z_{ik} \rangle + \text{Linear}_h(z_{jk}) \right), \tag{76}$$

$$\tilde{z}_{ij}^{(\text{start})} = g_{ij} \odot \text{Linear}\left(\sum_{k,h} A_{h,jk}^{(\text{start})} W_h^V z_{ik} \right). \tag{77}$$

3.4.5 Attenzione triangolare (ending node)

$$A_{h,ik}^{(\text{end})} = \text{softmax}_k\left(\frac{1}{\sqrt{d_h}} \langle W_h^Q z_{ij}, W_h^K z_{kj} \rangle + \text{Linear}_h(z_{ik}) \right). \tag{78}$$

Perché nessun LLM usa aggiornamenti triangolari

Il testo non ha distanze geometriche fra token. Lo spazio semantico è ad alta dimensione e non euclideo: "cane" non ha una distanza euclidea da "gatto". Solo nella modellazione di oggetti fisici emerge la necessità di rispettare la geometria euclidea, e con essa la consistenza triangolare. Gli aggiornamenti triangolari sono la *firma matematica* del dominio fisico nell'architettura.

3.5 §472–475 Lo Structure Module

3.5.1 Aggiornamento del backbone con quaternioni

$$T_i^{(t+1)} = T_i^{(t)} \circ \Delta T_i^{(t)}, \quad \Delta T_i^{(t)} = \text{BackboneUpdate}(s_i^{(t)}). \tag{79}$$

$$b_i = \text{Linear}(s_i) \in \mathbb{R}^6, \quad (R_i, t_i) = \text{QuatToMatrix}(b_i[1:4]) \oplus b_i[5:6]. \tag{80}$$

La formula di Rodrigues per i quaternioni $q = (q_0, q_1, q_2, q_3)$, $\|q\| = 1$:

$$R(q) = \begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}. \tag{81}$$

Osservazione 3.2 (Quaternioni vs angoli di Eulero)**.** La parametrizzazione di SO(3) tramite angoli di Eulero soffre del *gimbal lock*: quando $\beta = \pi/2$, α e γ diventano degeneri e il gradiente diverge. I quaternioni non hanno questa singolarità, rendendo l'ottimizzazione numericamente stabile.

3.6 §476 Invariant Point Attention (IPA): derivazione completa

L'IPA combina attenzione scalare standard con punti nello spazio 3D, garantendo SE(3)-invarianza.

Data la rappresentazione scalare $s_i \in \mathbb{R}^{c_s}$ e il frame $T_i = (R_i, t_i)$, l'IPA calcola punti 3D di query e key nel sistema di riferimento globale:

$$q_{i,hk}^{3D} = T_i \cdot (W_{hk}^{q,3D} s_i) = R_i W_{hk}^{q,3D} s_i + t_i. \tag{82}$$

Il punteggio IPA combina tre termini:

$$a_{ij}^{(h)} = w_s \langle W_h^Q s_i, W_h^K s_j \rangle - w_{3D} \sum_{k=1}^{N_{qk}} \left\| q_{i,hk}^{3D} - k_{j,hk}^{3D} \right\|^2 + \text{Linear}_h(z_{ij}). \tag{83}$$

Teorema 3.3 (Invarianza IPA a SE(3))**.** *Lo score $a_{ij}^{(h)}$ in (83) è invariante sotto qualsiasi trasformazione globale $(R_g, t_g) \in \text{SE}(3)$.*

Dimostrazione. Sotto (R_g, t_g): $q_{i,hk}^{3D} \mapsto R_g q + t_g$, $k_{j,hk}^{3D} \mapsto R_g k + t_g$. Allora $\|(R_g q + t_g) - (R_g k + t_g)\|^2 = \|R_g(q-k)\|^2 = \|q-k\|^2$ (isometria di R_g). Il termine scalare e il bias di coppia non dipendono dai frames. □ □

Output dell'IPA.

$$s_i \leftarrow s_i + \text{Linear}\left(\text{Concat}\left(\sum_j A_{ij} W^V s_j,\ \sum_j A_{ij} T_i^{-1} \cdot v_j^{3D},\ \sum_j A_{ij} \| T_i^{-1} \cdot v_j^{3D} \|^2 \right) \right), \tag{84}$$

dove $v_j^{3D} = W^{V,3D} s_j + t_j$ e le coordinate $T_i^{-1} \cdot v_j^{3D}$ sono nel frame locale — SE(3)-invarianti per la proposizione §460.

3.7 §477 Angoli torsionali e ricostruzione atomistica

AlphaFold predice sette angoli torsionali per residuo: (φ, ψ, ω) del backbone e $(\chi_1, \chi_2, \chi_3, \chi_4)$ delle catene laterali. Ogni angolo α è parametrizzato come vettore 2D normalizzato:

$$(s_\alpha, c_\alpha) = \text{normalize}(\text{Linear}(s_i)), \quad \alpha \in \{\varphi, \psi, \omega, \chi_1, \chi_2, \chi_3, \chi_4\}. \tag{85}$$

Osservazione 3.4 (Perché parametrizzazione 2D?)**.** La parametrizzazione scalare soffrirà di discontinuità a $\alpha = \pm\pi$. La perdita per due angoli quasi identici $\pi - \varepsilon$ e $-\pi + \varepsilon$ sarebbe $(2\pi)^2$. Con $(s, c) = (\sin\alpha, \cos\alpha)$: $\|(s_1, c_1) - (s_2, c_2)\|^2 = 2(1 - \cos(\alpha_1 - \alpha_2)) \approx (\alpha_1 - \alpha_2)^2$ per angoli vicini, eliminando la discontinuità.

La posizione di ogni atomo si ricostruisce per composizione di trasformazioni SE(3):

$$r_a = T_i \circ T_{\chi_1}^{(i)} \circ \cdots \circ T_{\chi_k}^{(i)} \cdot r_a^{(\text{ideal})}. \tag{86}$$

3.8 §478 La loss FAPE

Definizione 3.5 (FAPE loss)**.**

$$\text{FAPE}(\{\hat{T}_i\}, \{\hat{x}_a\}, \{T_i\}, \{x_a\}) = \frac{1}{L \cdot N_{\text{atoms}}} \sum_{i=1}^{L} \sum_{a=1}^{N_{\text{atoms}}} \left\| T_i^{-1} \cdot \hat{x}_a - T_i^{-1} \cdot x_a \right\|_c, \tag{87}$$

dove $\| \cdot \|_c = \min(\| \cdot \|, \varepsilon_c)$ con $\varepsilon_c \sim 10$ Å.

Proposizione 3.6 (Invarianza SE(3) della FAPE)**.** *La FAPE è invariante sotto trasformazioni globali* SE(3).

Dimostrazione. Sia $x'_a = R_g x_a + t_g$, $T'_i = (R_g R_i, R_g t_i + t_g)$. Allora $(T'_i)^{-1} x'_a = R_i^\top R_g^\top (R_g x_a + t_g - R_g t_i - t_g) = T_i^{-1} x_a$. □ □

Gradiente della FAPE.

$$\frac{\partial \mathcal{L}_{\text{FAPE}}}{\partial \hat{x}_a} = \frac{1}{LN_{\text{atoms}}} \sum_i \frac{T_i^{-\top}(T_i^{-1}(\hat{x}_a - x_a))}{\|T_i^{-1}(\hat{x}_a - x_a)\| + \varepsilon}. \tag{88}$$

È la somma di versori unitari (in coordinate locali) puntanti dall'atomo predetto verso quello vero: un segnale geometrico nel senso più letterale.

Nota. La differenza critica con la loss di un LLM: la FAPE opera in $\mathbb{R}^3$, si misura in Ångström. La cross-entropy di un LLM misura la sorpresa in bit. Non c'è unità fisica, non c'è geometria euclidea, non c'è norma ℓ_2 nello spazio dell'output linguistico.

Loss totale di AlphaFold 2.

$$\mathcal{L}_{\text{AF2}} = \mathcal{L}_{\text{FAPE}}^{(\text{bb})} + \mathcal{L}_{\text{FAPE}}^{(\text{sc})} + w_{\text{ang}}\mathcal{L}_{\text{ang}} + w_\chi \mathcal{L}_\chi + w_{\text{dist}}\mathcal{L}_{\text{dist}} + w_{\text{conf}}\mathcal{L}_{\text{conf}}. \tag{89}$$

3.9 §479 pLDDT, pTM e recycling

3.9.1 pLDDT: confidenza per residuo

$$\text{LDDT}_i = \frac{1}{|N_i|} \sum_{j \in N_i} \frac{1}{4} \sum_{\tau \in \{0.5,1,2,4\}} \mathbf{1}\Big[|d_{ij} - \hat{d}_{ij}| < \tau\Big], \tag{90}$$

dove N_i è l'insieme dei residui entro 15 Å da i nella struttura vera. Il pLDDT è predetto come classificatore a 50 bin; la loss di confidenza è una cross-entropy.

3.9.2 pTM score: confidenza globale

$$\text{pTM} = \max_i \frac{1}{L} \sum_j \sum_{b=1}^{64} p_\theta(\text{bin}_b \mid z_{ij}) \cdot \frac{1}{1 + (d_b/d_0(L))^2}, \quad d_0(L) = 1.24(L-15)^{1/3} - 1.8 \text{ Å}. \tag{91}$$

3.9.3 Recycling come punto fisso

$$(s^{(0,t+1)}, z^{(0,t+1)}) = \text{Recycle}\Big(s^{(N_{\text{evo}},t)}, z^{(N_{\text{evo}},t)}, \{T_i^{(t)}\}\Big), \quad t = 0,1,2. \tag{92}$$

L'intuizione: si cerca un punto fisso della mappa $(s, z) \mapsto (s', z')$. Se la struttura predetta è corretta, il modello non la modificherà nell'iterazione successiva. Il numero standard è 3 iterazioni di recycling.

Proposizione 3.7 (Condizione di punto fisso)**.** *(s^*, z^*) è un punto fisso della mappa Evoformer+StructureModule se e solo se ogni aggiornamento del backbone $\Delta T_i^* = \text{BackboneUpdate}(s_i^*)$ è l'identità in* SE(3)*.*

3.10 §480 AlphaFold 3: diffusione su strutture molecolari

AlphaFold 3 (Abramson et al., *Nature* 2024) estende AF2 a strutture molecolari generali: proteine, DNA, RNA, ligandi, ioni e modificazioni post-traduzionali. La principale innovazione matematica è la sostituzione dello Structure Module con un *modello di diffusione condizionale.*

3.10.1 Il Pairformer

L'Evoformer di AF2 viene sostituito dal *Pairformer*, che opera su token eterogenei (residui proteici, nucleotidi, atomi del ligando) con una rappresentazione di coppia $z_{ij}^{\text{AF3}} \in \mathbb{R}^{c_z}$ definita su tutti i token.

3.10.2 Loss di diffusione condizionale

Si parte dalle coordinate atomiche vere $x_0 \in \mathbb{R}^{3N}$ e si costruisce il processo di noising:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{93}$$

dove $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. Il modello di denoising $D_\theta(x_t, t, c)$, condizionato sulla rappresentazione (s, z) del Pairformer, predice x_0:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t,x_0,\epsilon}\left[\lambda(t)\, \|D_\theta(x_t, t, c) - x_0\|^2\right]. \tag{94}$$

Osservazione 3.8. L'invarianza SE(3) è ottenuta facendo predire a D_θ le coordinate in un sistema di riferimento canonico (estratto dalle coordinate rumorose tramite un frame globale) e poi applicando il frame inverso. Questo approccio è più flessibile dell'IPA: non richiede una struttura di frame rigidi pre-definita e si adatta a molecole con topologie arbitrarie.

Tabella 9: AlphaFold 2 vs AlphaFold 3: differenze architetturali

Aspetto	**AlphaFold 2**	**AlphaFold 3**
Dominio	Solo proteine	Proteine, DNA, RNA, ligandi
Encoder evolutivo	Evoformer + MSA	Pairformer (MSA opzionale)
Decoder strutturale	Structure Module (IPA)	Modello di diffusione
Loss strutturale	FAPE (deterministica)	$\mathcal{L}_{\text{diff}}$ (probabilistica)
Output	Struttura singola	Ensemble di strutture
Parametri	$\sim$ 93M	$\sim$ 700M

3.11 §472a Backbone update con quaternioni: derivazione completa

La rappresentazione dei frames con matrici di rotazione $R \in \mathrm{SO}(3)$ ha dimensione 9 ma gradi di libertà effettivi 3. I *quaternioni unitari* offrono una parametrizzazione minimale e numericamente stabile.

3.11.1 Algebra dei quaternioni

Un quaternione $q = q_0 + q_1 i + q_2 j + q_3 k \in \mathbb{H}$ con $\|q\| = 1$ corrisponde a una rotazione di angolo $2\arccos(q_0)$ attorno all'asse $(q_1, q_2, q_3)/\sin(\arccos(q_0))$. La rotazione applicata a un vettore $v \in \mathbb{R}^3$ (rappresentato come quaternione puro $v = 0 + v_1 i + v_2 j + v_3 k$) è:

$$v' = qvq^{-1} = qv\bar{q}, \tag{95}$$

dove $\bar{q} = q_0 - q_1 i - q_2 j - q_3 k$ è il coniugato (equivale a q^{-1} per quaternioni unitari).

Da quaternione a matrice di rotazione.

$$R(q) = \begin{pmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1-2(q_1^2+q_2^2) \end{pmatrix}. \tag{96}$$

Composizione di rotazioni. La composizione $R_1 \circ R_2$ corrisponde al prodotto quaternionico $q_1 q_2$ (non commutativo). Questo permette di costruire il backbone update come composizione di trasformazioni locali senza mai materializzare la matrice 3×3.

3.11.2 L'aggiornamento iterativo nel Structure Module

Lo Structure Module aggiorna i frames in $N_{\text{str}} = 8$ strati. Ad ogni strato t:

$$T_i^{(t+1)} = T_i^{(t)} \circ \Delta T_i^{(t)}, \tag{97}$$

dove $\Delta T_i^{(t)} = (\Delta R_i^{(t)}, \Delta t_i^{(t)})$ è la trasformazione incrementale predetta dalla rete dal residuo i allo strato t.

Predizione del delta. La trasformazione incrementale viene predetta da una testa lineare applicata alle feature di sequenza:

$$a_i^{(t)} = \text{LayerNorm}(\text{IPA}(s_i^{(t)}, z_{ij}, T_i^{(t)})), \tag{98}$$

$$s_i^{(t+1)} = \text{Transition}(a_i^{(t)}), \tag{99}$$

$$(b_i, c_i) = \text{Linear}(s_i^{(t+1)}), \quad b_i \in \mathbb{R}^7, c_i \in \mathbb{R}^3, \tag{100}$$

$$\Delta q_i = \text{normalize}(b_i[0:4]), \quad \Delta t_i = b_i[4:7], \tag{101}$$

$$\Delta T_i^{(t)} = (\Delta q_i, \Delta t_i). \tag{102}$$

Proposizione 3.9 (Invarianza della composizione di frames)**.** *La formula* (97) *è* SE(3)*-equivariante: se il frame iniziale $T_i^{(0)}$ viene trasformato da (R_g, t_g), tutti i frame successivi $T_i^{(t)}$ vengono trasformati coerentemente da (R_g, t_g).*

Dimostrazione. Per induzione su t. Il caso base $T_i^{(0)}$ è trasformato per ipotesi. Assumendo $T_i^{(t)} = (R_g R_i^{(t)}, R_g t_i^{(t)} + t_g)$, le feature $s_i^{(t)}$ sono invarianti (IPA è invariante). Quindi $\Delta T_i^{(t)}$ non dipende da (R_g, t_g). La composizione:

$$T_i^{(t+1)} = (R_g R_i^{(t)} \Delta R_i^{(t)},\ R_g R_i^{(t)} \Delta t_i^{(t)} + R_g t_i^{(t)} + t_g) = (R_g R_i^{(t+1)},\ R_g t_i^{(t+1)} + t_g). \quad \square$$

□

3.11.3 Violazione dei legami e stop-gradient

Un problema pratico è che le strutture generate dallo Structure Module possono violare i vincoli geometrici del backbone (distanze di legame, angoli planari). AlphaFold usa due soluzioni:

1. **Loss di violazione di legame**: si aggiunge alla loss totale una penalità quadratica sulle violazioni di distanza:

$$\mathcal{L}_{\text{viol}} = \sum_i \text{ReLU}(d_{i,i+1} - d_{\max})^2 + \text{ReLU}(d_{\min} - d_{i,i+1})^2, \tag{103}$$

   dove $d_{i,i+1} = \|C_i - N_{i+1}\|$ è la distanza del legame peptidico ($d_{\text{ideal}} \approx 1.33$ Å).

2. **Stop-gradient selettivo**: durante il training, il gradiente non fluisce attraverso le trasformazioni di composizione dei frames per evitare instabilità nelle prime fasi. È rilasciato progressivamente con l'aumentare del numero di passi di recycling.

3.12 §473a Distillazione e training su strutture predette

AlphaFold 2 è addestrato non solo su strutture cristallografiche (PDB), ma anche su strutture predette da versioni precedenti di AlphaFold stesso. Questa tecnica di *knowledge distillation* strutturale ha permesso di usare dati senza struttura sperimentale nota.

Definizione 3.10 (Distillation loss strutturale)**.** Siano $\hat{T}_i^{\text{teacher}}$ le strutture predette dal modello teacher. La loss di distillazione è:

$$\mathcal{L}_{\text{distil}} = \lambda_{\text{d}}\,\text{FAPE}\Big(\hat{T}_i^{\text{student}}, \hat{x}_a^{\text{student}},\ \hat{T}_i^{\text{teacher}}, \hat{x}_a^{\text{teacher}}\Big)\,. \tag{104}$$

Il modello student impara a predire strutture simili al teacher, anche senza struttura sperimentale di riferimento.

Attenzione

La distillazione introduce un rischio di *bias da teacher*: se il teacher commette errori sistematici su certi fold, il student li apprende. AlphaFold mitiga questo usando il pLDDT del teacher come peso: le regioni con bassa confidenza del teacher ricevono peso minore nella loss di distillazione.

3.13 §474b Il diagramma di Ramachandran: vincoli geometrici del backbone

Il piano del legame peptidico è *rigido*: l'angolo ω tra C^{α}_{i-1}-C_{i-1}-N_i-C^{α}_i è tipicamente $\omega \approx 180$ (configurazione *trans*) a causa della natura parzialmente doppia del legame C-N. I gradi di libertà effettivi del backbone si riducono quindi agli angoli di Ramachandran (φ_i, ψ_i) per ogni residuo i.

3.13.1 Definizione degli angoli di Ramachandran

Gli angoli sono definiti sulle quattro posizioni atomiche del backbone:

$$\varphi_i = \text{dihedral}(C_{i-1},\ N_i,\ C^{\alpha}_i,\ C_i), \tag{105}$$
$$\psi_i = \text{dihedral}(N_i,\ C^{\alpha}_i,\ C_i,\ N_{i+1}), \tag{106}$$

dove $\text{dihedral}(a,b,c,d)$ è l'angolo tra i piani (a,b,c) e (b,c,d).

3.13.2 La distribuzione di Ramachandran come prior fisico

Le strutture proteiche naturali occupano solo una piccola parte dello spazio $(\varphi,\psi) \in [-\pi,\pi]^2$. Le regioni favorevoli corrispondono a strutture secondarie canoniche:

- α-elica: $(\varphi,\psi) \approx (-57,-47)$;
- β-sheet parallelo: $(\varphi,\psi) \approx (-119,+113)$;
- β-sheet antiparallelo: $(\varphi,\psi) \approx (-139,+135)$.

AlphaFold 2 incorpora implicitamente questo vincolo attraverso la loss sugli angoli:

$$\mathcal{L}_{\text{ang}} = \frac{1}{L}\sum_{i=1}^{L}\sum_{\alpha\in\{\varphi,\psi,\omega\}} \big\|(\sin\hat{\alpha}_i,\cos\hat{\alpha}_i) - (\sin\alpha_i^*,\cos\alpha_i^*)\big\|^2\,, \tag{107}$$

dove $(\sin\hat{\alpha},\cos\hat{\alpha})$ è la predizione 2D (85) e $(\sin\alpha^*,\cos\alpha^*)$ è il ground truth.

Proposizione 3.11 (Equivalenza circolare della loss angolare)**.** *La loss (107) è equivalente a $2(1-\cos(\hat{\alpha}-\alpha^*))$, che è la metrica della distanza circolare al quadrato (approssimazione per angoli vicini all'angolo di differenza $|\hat{\alpha}-\alpha^*|^2$). Non ha discontinuità a $\pm\pi$, al contrario della MSE su angoli scalari.*

3.13.3 Confronto con i LLM

Nei LLM non esiste un equivalente del diagramma di Ramachandran: la distribuzione dei token non ha una geometria nello spazio $[-\pi, \pi]^2$ con regioni "vietate" dalla fisica. La distribuzione dei token è statisticamente appresa dal corpus senza prior fisici. Questo riflette la differenza fondamentale: le proteine devono soddisfare vincoli fisico-chimici stringenti che derivano dalla meccanica quantistica dei legami, mentre il testo segue vincoli grammaticali e semantici che sono convenzionali e variano tra lingue e stili.

3.14 §480b Score Matching e la connessione con i modelli di diffusione in AF3

La diffusione condizionale in AlphaFold 3 (93) è un'istanza del framework DDPM (Denoising Diffusion Probabilistic Models). Deriviamo il suo fondamento matematico dal *score matching*.

3.14.1 Score function e stima

Lo *score* di una distribuzione $p(x)$ è il gradiente del log-likelihood:

$$s(x) = \nabla_x \log p(x). \tag{108}$$

La *score matching* (Hyvärinen, 2005) stima $s(x)$ minimizzando:

$$J_{\mathrm{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{p(x)}\left[\|s_\theta(x) - \nabla_x \log p(x)\|^2\right]. \tag{109}$$

Il problema è che $\nabla_x \log p(x)$ non è noto. Usando l'identità di integrazione per parti (Stein's identity):

$$J_{\mathrm{SM}}(\theta) = \mathbb{E}_{p(x)}\left[\mathrm{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|^2\right] + \mathrm{const}, \tag{110}$$

minimizzabile senza accesso a $p(x)$.

3.14.2 Denoising Score Matching (DSM) e DDPM

La *Denoising Score Matching* (Vincent, 2011) aggira il calcolo della traccia Jacobiana in (110) usando dati corrotti:

$$J_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\lambda(t)\left\|s_\theta(x_t, t) - \frac{-\epsilon}{\sqrt{1-\bar{\alpha}_t}}\right\|^2\right], \tag{111}$$

dove $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$. La relazione tra lo score stimato e il denoiser è:

$$s_\theta(x_t, t) = -\frac{\epsilon_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}, \tag{112}$$

dove ϵ_θ è la rete che predice il rumore (parametrizzazione ϵ-prediction) o equivalentemente $D_\theta(x_t, t) = (x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta)/\sqrt{\bar{\alpha}_t}$ (parametrizzazione x_0-prediction, usata in AF3).

Proposizione 3.12 (Equivalenza delle parametrizzazioni)**.** *Minimizzare $\mathcal{L}_{diff}$ (94) (parametrizzazione x_0-prediction) è equivalente a minimizzare (111) (parametrizzazione ϵ-prediction), con la relazione:*

$$D_\theta(x_t, t) = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}. \tag{113}$$

Le due loss differiscono solo per un riscalamento dipendente da t.

3.14.3 Il processo di campionamento: DDPM vs DDIM

Data una stima D_θ del denoiser, il campionamento da $p(x_0)$ avviene risolvendo l'equazione di reverse diffusion. In DDPM (stochastic):

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right) + \sigma_t z, \quad z \sim \mathcal{N}(0, I), \tag{114}$$

dove $\sigma_t^2 = (1-\alpha_t)(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)$. In DDIM (deterministic, Song et al. 2020), si impone $\sigma_t = 0$, ottenendo un ODE:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} D_\theta(x_t, t) + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \frac{x_t - \sqrt{\bar{\alpha}_t} D_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}. \tag{115}$$

DDIM permette il campionamento con 10–50 passi invece di 1000, cruciale per le applicazioni di inferenza proteica di AF3.

3.15 §480c Flow Matching: un'alternativa più stabile alla diffusione

Conditional Flow Matching (Lipman et al., 2022; Albergo e Vanden-Eijnden, 2022) è un framework di training più semplice e stabile rispetto a DDPM, adottato da modelli recenti come RFdiffusion e Boltz-1.

3.15.1 Flussi di probabilità condizionali

Un *Continuous Normalizing Flow* (CNF) trasporta una distribuzione semplice $p_0 = \mathcal{N}(0, I)$ a una distribuzione target $p_1 \approx p_{\text{data}}$ tramite un campo vettoriale $v_t(x) : [0,1] \times \mathbb{R}^n \to \mathbb{R}^n$:

$$\frac{dx_t}{dt} = v_t(x_t), \quad x_0 \sim p_0, \quad x_1 \sim p_1. \tag{116}$$

L'obiettivo di training è apprendere $v_\theta \approx v_t$. La difficoltà è che v_t non è accessibile direttamente.

3.15.2 Conditional Flow Matching (CFM)

CFM risolve il problema usando un percorso *condizionale* $p_t(x \mid x_1)$:

$$p_t(x \mid x_1) = \mathcal{N}(x;\ \mu_t(x_1),\ \sigma_t(x_1)^2 I), \tag{117}$$

con $\mu_0(x_1) = 0$, $\sigma_0 = 1$ (distribuzione iniziale) e $\mu_1(x_1) = x_1$, $\sigma_1 \to 0$ (distribuzione target). La scelta "lineare" semplice è $\mu_t(x_1) = t x_1$, $\sigma_t = 1 - t$, producendo:

$$x_t = t\, x_1 + (1-t)\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{118}$$

Il campo vettoriale condizionale corrispondente è:

$$u_t(x \mid x_1) = \frac{x_1 - x_t}{1-t} = \frac{x_1 - \epsilon}{1}, \quad \text{(indipendente da } t\text{)}. \tag{119}$$

Teorema 3.13 (CFM loss = regression sul campo vettoriale)**.** *Minimizzare la CFM loss:*

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t, x_0, x_1}\left[\|v_\theta(x_t, t) - u_t(x_t \mid x_1)\|^2\right] \tag{120}$$

è equivalente a minimizzare la distanza al campo vettoriale marginale $v_t(x)$, che corrisponde al campo di flusso ottimale di trasporto.

Tabella 10: Confronto DDPM vs Flow Matching per strutture molecolari

Aspetto	**DDPM (AF3)**	**Flow Matching (Boltz-1)**
Training loss	$\|\epsilon_\theta - \epsilon\|^2$	$\|v_\theta - u_t\|^2$
Schedule noise	β_t schedule (fisso)	Interpolazione lineare
Campionamento	SDE/ODE, $T = 1000$	ODE, $T = 50$
Stabilità	Richieede β_t scheduling	Stabile per costruzione
Invarianza SE(3)	Frame canonico	Frame canonico

4 Sezione D — AlphaGenome: La Frontiera Genomica

4.1 §481 Motivazione e dominio

AlphaGenome (Google DeepMind, 2025) estende l'approccio di AlphaFold al genoma: il suo obiettivo è predire la *funzione genomica* — espressione genica, accessibilità della cromatina, siti di legame dei fattori di trascrizione, struttura 3D del cromosoma — a partire dalla sequenza di DNA.

Tabella 11: Confronto dei domini di input

Proprietà	**AlphaFold (proteine)**	**AlphaGenome (genoma)**
Lunghezza input	50–2500 residui	fino a 1 Mbp (10^6 basi)
Alfabeto	20 amminoacidi	4 nucleotidi (A, C, G, T)
Output primario	Coordinate 3D ($\mathbb{R}^3$)	Segnali funzionali (multi-task)
Simmetria rilevante	SE(3)	Complementarità WC, strand
Scala temporale	Proprietà stazionaria	Dipendente da tipo cellulare
Confronto con LLM	Sequenza 1D + MSA 2D	Sequenza 1D multi-scala

La sfida della scala genomica. Il genoma umano ha $\sim 3 \times 10^9$ basi. Un LLM che processasse l'intero genoma richiederebbe una finestra di contesto di ~ 750 milioni di token (con 4 basi per token), computazionalmente impossibile con attenzione quadratica $O(N^2)$. AlphaGenome risolve questo con una gerarchia multi-scala.

I fenomeni genomici hanno scale caratteristiche radicalmente diverse:

- motivo di legame di un fattore di trascrizione: ~ 10 bp;
- promotore di un gene: ~ 200 bp;
- gene tipico: $\sim 10^4$ bp;
- dominio di associazione topologica (TAD): $\sim 10^6$ bp.

Nessuna architettura a risoluzione singola (né LLM né AlphaFold) può modellare simultaneamente queste scale in modo efficiente.

4.2 §482 Tokenizzazione genomica multi-scala

A differenza dei LLM (che usano BPE sul testo) e di AlphaFold (che usa singoli amminoacidi), AlphaGenome adotta una rappresentazione gerarchica del DNA.

4.2.1 Tokenizzazione al livello di nucleotide

Ogni base è codificata come vettore one-hot $e_b \in \mathbb{R}^4$ per $b \in \{A, C, G, T\}$. Per una regione genomica di L_0 basi, l'input è $X^{(0)} \in \mathbb{R}^{L_0 \times 4}$.

4.2.2 Stem convolutivo multi-scala

La riduzione di scala è implementata con una torre di convoluzioni 1D con pooling:

$$X^{(l)} = \mathrm{Conv1D}(X^{(l-1)})_{\downarrow r_l}, \quad l = 1, \ldots, L_{\mathrm{stem}}, \tag{121}$$

dove $(\cdot)_{\downarrow r_l}$ indica il downsampling di un fattore r_l. Con fattori tipici $r_l \in \{2, 2, 2, 2, 4\}$ e $L_{\mathrm{stem}} = 5$, la risoluzione passa da $L_0 \approx 10^6$ a $L_0/128 \approx 8000$ token. Ogni livello usa:

$$\mathrm{Conv1D}_l(x)_i = \mathrm{GELU}\left(\sum_{j=0}^{k_l - 1} W_l^{(j)} x_{i+j} + b_l \right), \tag{122}$$

seguito da max-pooling o attention-pooling per il downsampling.

4.2.3 Livelli di risoluzione multipla

AlphaGenome mantiene rappresentazioni a quattro risoluzioni contemporaneamente (analoga a una piramide gaussiana):

$$\left\{ X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)} \right\} \text{ con risoluzioni } \{16 \text{ bp}, 128 \text{ bp}, 1 \text{ kbp}, 8 \text{ kbp}\}. \tag{123}$$

Nota. Questo è fondamentalmente diverso da tutti gli LLM e da AlphaFold, che operano a risoluzione singola. La necessità di più scale non è arbitraria: è imposta dalla struttura fisica del genoma, in cui fenomeni a 10 bp influenzano fenomeni a 10^6 bp.

4.3 §483–486 Attenzione cross-scala

Il cuore computazionale di AlphaGenome è un meccanismo di *attenzione cross-scala* che permette a rappresentazioni a risoluzioni diverse di scambiarsi informazione.

Siano $X^{(\ell_1)} \in \mathbb{R}^{N_1 \times d}$ e $X^{(\ell_2)} \in \mathbb{R}^{N_2 \times d}$ le rappresentazioni a due risoluzioni diverse, con $N_1 > N_2$ (cioè $\ell_1 < \ell_2$, risoluzione più alta). L'attenzione cross-scala da ℓ_2 verso ℓ_1 è:

$$Q^{(\ell_2 \to \ell_1)} = X^{(\ell_2)} W^Q, \quad K^{(\ell_1)} = X^{(\ell_1)} W^K, \quad V^{(\ell_1)} = X^{(\ell_1)} W^V, \tag{124}$$

$$X^{(\ell_2)}_{\mathrm{updated}} = X^{(\ell_2)} + \mathrm{Attn}\Big(Q^{(\ell_2 \to \ell_1)}, K^{(\ell_1)}, V^{(\ell_1)} \Big). \tag{125}$$

Gestione del mismatch dimensionale. La dimensione $N_1 \neq N_2$ è gestita da un meccanismo di interpolazione posizionale: per ogni posizione genomica p_2 alla risoluzione ℓ_2, si calcola il contributo da tutti i token p_1 alla risoluzione ℓ_1 che si sovrappongono alla finestra genomica corrispondente a p_2, con pesi di attenzione che decadono con la distanza genomica.

4.3.1 Position encoding genomico

Per l'attenzione intra-scala, AlphaGenome usa RoPE adattato con frequenze base scelte per coprire il range dinamico delle scale genomiche. Per l'attenzione cross-scala, si usa un encoding di distanza genomica relativa in base logaritmica:

$$\mathrm{PE}_{\log}(d) = \mathrm{Embedding}(\lfloor \log_2(d + 1) \rfloor), \quad d = |p_1 \cdot r_{\ell_1} - p_2 \cdot r_{\ell_2}| \text{ (in bp)}, \tag{126}$$

dove r_ℓ è la risoluzione in bp al livello ℓ.

Motivazione della base logaritmica. La discretizzazione logaritmica è motivata dall'invarianza di scala dei meccanismi epigenetici: un potenziatore distante 10 kbp ha un effetto simile a uno distante 9 kbp o 11 kbp, ma molto diverso da uno distante 1 Mbp. La mappa $d \mapsto \lfloor \log_2(d+1) \rfloor$ comprime uniformemente l'asse delle distanze in modo proporzionale al rapporto tra distanze, non alla differenza assoluta.

Tabella 12: Confronto tra i position encoding nei tre sistemi

Aspetto	**LLM**	**AlphaFold**	**AlphaGenome**
Tipo	RoPE / ALiBi / sin	Posizione discreta + pairwise	Log-distanza genomica
Spazio	Posizione sequenziale	Sequenza + 3D implicito	Multi-scala (4 livelli)
Range	$[0, T_{\max}]$	$[0, L]$	[1 bp, 1 Mbp]
Linearità	Lineare / logaritmica	Lineare	Logaritmica
Cross-risoluzione	No	No	Sì (attenzione cross-scala)

4.4 §487–490 Complementarità di Watson-Crick come simmetria

Il DNA è doppio-filamento: ogni base b sul filamento diretto ha una base complementare $\bar{b}$ sul filamento inverso ($A \leftrightarrow T$, $C \leftrightarrow G$). La trascrizione può avvenire su entrambi i filamenti.

4.4.1 Formalizzazione matematica

Sia $\varphi : \{A, C, G, T\}^L \to \{A, C, G, T\}^L$ l'operatore di *complemento-inversione* (reverse complement):

$$\varphi(b_1 b_2 \cdots b_L) = \bar{b}_L \bar{b}_{L-1} \cdots \bar{b}_1. \tag{127}$$

La complementarità di Watson-Crick implica che $f(\varphi(x))$ e $f(x)$ debbano predire le stesse proprietà funzionali (a meno di un ribaltamento del filamento): la rete deve essere $\mathbb{Z}_2$-*equivariante* rispetto a φ.

4.4.2 Implementazione tramite weight sharing e data augmentation

AlphaGenome garantisce l'equivarianza a φ tramite:

1. **Weight sharing**: i pesi della rete sono condivisi tra il processamento del filamento diretto e di quello inverso;
2. **Media delle predizioni**:
$$\hat{y} = \frac{1}{2}\left[f(x) + f(\varphi(x))\right]. \tag{128}$$

Tabella 13: Confronto degli approcci all'equivarianza

Modello	**Gruppo**	**Metodo**	**Costo extra**	**Esattezza**
AlphaFold IPA	SE(3)	Frames locali	$O(L^2)$	Esatta
AlphaGenome WC	$\mathbb{Z}_2$	Test-time aug.	$2\times$ inferenza	Esatta
e3nn	SE(3)	Armoniche sferic.	$O(L^3)$	Esatta
Data augmentation	Qualsiasi G	Campionamento	$\lvert G\rvert\times$	Approssimata

Confronto con l'approccio IPA di AlphaFold.

Nota. La simmetria Watson-Crick non ha equivalente nei LLM (il testo non ha complemento) né in AlphaFold (la sequenza proteica non ha complemento). È una simmetria specifica al DNA, codificata nel modello con una tecnica semplice e potente — test-time augmentation e weight tying — invece delle architetture equivarianti sofisticate come IPA.

4.5 §491–494 Teste di predizione multi-task

AlphaGenome predice simultaneamente decine di fenotipi molecolari con una singola rete condivisa e teste specifiche.

4.5.1 Head per l'espressione genica (RNA-seq)

Per ogni tipo cellulare c e gene g, la predizione del livello di espressione (in RPKM) è:

$$\hat{y}_{g,c}^{(\text{expr})} = \text{softplus}\Big(W_c^{(\text{expr})} \text{Pool}(X_g^{(\ell_g)}) + b_c^{(\text{expr})}\Big), \tag{129}$$

dove $\text{Pool}(\cdot)$ è un'operazione di pooling sul gene g alla risoluzione ℓ_g appropriata e $\text{softplus}(x) = \log(1 + e^x)$ garantisce positività.

Motivazione della softplus. L'espressione genica è una grandezza non negativa (conti di read), con distribuzione altamente asimmetrica (pochi geni molto espressi, molti geni poco espressi). La softplus approssima ReLU per $x \gg 0$ ma è differenziabile in $x = 0$, evitando il "dying neuron" problem tipico di ReLU per output di predizione.

4.5.2 Head per l'accessibilità della cromatina (ATAC-seq)

L'accessibilità misura se una regione è aperta (accessibile ai fattori di trascrizione) o chiusa (avvolta attorno ai nucleosomi):

$$\hat{y}_{i,c}^{(\text{atac})} = \sigma\Big(W_c^{(\text{atac})} x_i^{(\ell_1)} + b_c^{(\text{atac})}\Big), \tag{130}$$

dove $x_i^{(\ell_1)}$ è la rappresentazione alla risoluzione più fine (16 bp). La sigmoid mappa l'output in $[0, 1]$, interpretabile come probabilità di apertura.

4.5.3 Head per la struttura 3D del cromosoma (Hi-C)

L'esperimento Hi-C misura la frequenza di contatto tra regioni genomiche distanti, producendo una matrice $C \in \mathbb{R}^{L/r \times L/r}$. AlphaGenome predice questa matrice con una *testa bilineare*:

$$\hat{C}_{ij} = \text{softplus}\Big(\text{Linear}\Big(x_i^{(\ell_3)} \otimes x_j^{(\ell_3)}\Big)\Big), \tag{131}$$

una struttura che ricorda l'outer product mean dell'Evoformer (71), ma usata come testa di output anziché come aggiornamento interno.

Nota. La predizione della matrice Hi-C è matematicamente la struttura più simile ad AlphaFold tra tutti i task di AlphaGenome: entrambi modellano interazioni a coppie in uno spazio $L \times L$. La differenza è che AlphaFold usa queste interazioni per ricostruire coordinate 3D, mentre AlphaGenome le predice direttamente come output funzionale.

4.5.4 Head per i siti di legame dei fattori di trascrizione (ChIP-seq)

Per ogni fattore di trascrizione f e tipo cellulare c:

$$\hat{y}_{i,f,c}^{(\text{chip})} = \sigma\Big(W_{f,c}^{(\text{chip})} x_i^{(\ell_1)} + b_{f,c}^{(\text{chip})}\Big), \tag{132}$$

dove la sigmoid produce una probabilità di legame per ogni posizione genomica i.

4.5.5 Head per lo splicing alternativo

Lo splicing alternativo determina quali esoni vengono inclusi nel trascritto maturo. AlphaGenome predice la *percentage spliced in* (PSI) per ogni sito di splicing:

$$\hat{\Psi}_{i,c}^{(\text{splicing})} = \sigma\Big(W_c^{(\text{spl})}\text{Concat}(x_{i-\delta}^{(\ell_1)}, \ldots, x_{i+\delta}^{(\ell_1)}) + b_c^{(\text{spl})}\Big), \tag{133}$$

dove δ è la finestra contestuale attorno al sito di splicing.

4.6 §494 Funzione di perdita multi-task

$$\mathcal{L}_{\text{AG}} = \sum_{t \in \mathcal{T}} \lambda_t \mathcal{L}_t, \tag{134}$$

dove $\mathcal{T}$ include espressione genica, accessibilità ATAC, ChIP-seq, Hi-C, splicing, ecc. Le perdite $\mathcal{L}_t$ sono:

- **Pearson correlation loss** per segnali continui:

$$\mathcal{L}_{\text{Pearson}} = 1 - \frac{\text{Cov}(\hat{y}, y)}{\sigma_{\hat{y}}\sigma_y}; \tag{135}$$

- **Cross-entropy** per segnali binari (presenza/assenza di picco):

$$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i); \tag{136}$$

- **Poisson log-likelihood** per conteggi di read di sequenziamento:

$$\mathcal{L}_{\text{Poisson}} = -\sum_i \left(y_i \log \hat{y}_i - \hat{y}_i\right); \tag{137}$$

- **MSE pesato** per la matrice Hi-C in spazio logaritmico:

$$\mathcal{L}_{\text{Hi-C}} = \frac{1}{L^2} \sum_{i,j} w_{ij} (\log \hat{C}_{ij} - \log C_{ij})^2. \tag{138}$$

Motivazione della Pearson loss. La correlazione di Pearson è invariante a scala e offset: se tutti i valori predetti vengono moltiplicati per una costante, la correlazione non cambia. Questo la rende robusta alle differenze sistematiche di calibrazione tra esperimenti diversi, problema frequente nei dati di espressione genica.

Motivazione della Poisson loss. I dati di sequenziamento sono conteggi di read che seguono approssimativamente una distribuzione di Poisson: $y_i \sim \text{Poisson}(\hat{y}_i)$. La log-likelihood di Poisson è $\sum_i (y_i \log \hat{y}_i - \hat{y}_i) + \text{const}$, che massimizzata coincide con la minimizzazione di (137). Rispetto alla MSE, la Poisson loss penalizza proporzionalmente all'inverso della media: gli errori su geni poco espressi ricevono peso maggiore.

4.7 §495 Confronto matematico AlphaFold vs AlphaGenome

Confronto con Enformer e Basenji. AlphaGenome si inserisce in un continuum di modelli per la genomica funzionale:

- **Basenji** (Kelley et al., 2018): CNN pura, risoluzione singola di 128 bp, contesto di 131 kbp. Prima dimostrazione di predizione multi-task genomica a bassa risoluzione.

Tabella 14: AlphaFold 2 vs AlphaGenome: differenze matematiche fondamentali

Aspetto	**AlphaFold 2**	**AlphaGenome**
Input	Seq. amminoacidica + MSA	Seq. di DNA (fino a 1 Mbp)
Risoluzione	1 token = 1 residuo	4 scale: 16 bp–8 kbp
Simmetria	SE(3) (continua)	Complementarità WC ($\mathbb{Z}_2$, discreta)
Repr. coppia	$N \times N \times c_z$ (Evoformer)	Multi-scala: $N_1^2, N_2^2, \ldots$
Aggiornamenti triang.	Sì (4 varianti)	No (ma head bilineare Hi-C)
Output primario	Coordinate 3D	Segnali funzionali multi-task
Loss principale	FAPE (geometrica, in Å)	Multi-task (Pearson, Poisson, MSE)
Evolutivo	MSA esplicita	Non richiede allineamenti
Position encoding	Posizione discreta + pairwise	Log-distanza genomica
Decoder	IPA + angoli torsionali	Teste lineari/bilineari specifiche
Numero di task	1 (struttura)	> 50 (espressione, cromatina, 3D, ...)

- **Enformer** (Avsec et al., 2021): CNN + Transformer, risoluzione 128 bp, contesto di 196 kbp. Primo uso dell'attenzione nel dominio genomico, ma senza multi-scala.
- **AlphaGenome** (2025): architettura multi-scala completa, contesto di 1 Mbp, 4 risoluzioni, attenzione cross-scala. Prima modellazione della struttura 3D del cromosoma (Hi-C) in un modello unificato.

4.8 §492a Dinamiche del gradiente nel training multi-task

Il training multi-task di AlphaGenome ottimizza simultaneamente $|\mathcal{T}| > 50$ task. Questa complessità pone sfide matematiche specifiche che non compaiono nei training mono-task di LLM e AlphaFold.

4.8.1 Conflitto di gradiente

Due task t_1, t_2 sono in *conflitto di gradiente* quando:

$$\langle \nabla_\theta \mathcal{L}_{t_1},\ \nabla_\theta \mathcal{L}_{t_2} \rangle < 0, \tag{139}$$

cioè quando minimizzare t_1 peggiora t_2 (e viceversa). In presenza di conflitti sistematici, il gradiente aggregato $\sum_t \lambda_t \nabla_\theta \mathcal{L}_t$ può finire ortogonale a tutti i task individuali, rallentando enormemente la convergenza.

Frequenza empirica dei conflitti. Per una rete con 50 task genomici, il numero di coppie è $\binom{50}{2} = 1225$. In pratica, le coppie (espressione genica, accessibilità ATAC) mostrano conflitti ridotti (entrambi sono segnali di "regioni attive"), mentre (splicing, Hi-C) possono conflitti più forti (fenomeni su scale diverse).

4.8.2 PCGrad: proiezione per ridurre i conflitti

PCGrad (Yu et al., 2020) proietta il gradiente di ogni task sul piano ortogonale al gradiente di ogni task in conflitto:

$$\tilde{g}_t = g_t - \sum_{\substack{t' \neq t \\ \langle g_t, g_{t'} \rangle < 0}} \frac{\langle g_t, g_{t'} \rangle}{\|g_{t'}\|^2} g_{t'}, \tag{140}$$

dove $g_t = \nabla_\theta \mathcal{L}_t$.

Proposizione 4.1 (PCGrad riduce la perdita multi-task)**.** *Il gradiente aggregato $\sum_t \tilde{g}_t$ ottenuto con PCGrad soddisfa:*

$$\langle \sum_t \tilde{g}_t,\ g_{t_0} \rangle \geq 0 \quad \forall t_0 \in \mathcal{T},$$

cioè il passo in direzione $-\sum_t \tilde{g}_t$ non aumenta nessun task individuale (in prima approssimazione).

AlphaGenome e i pesi adattativi. AlphaGenome usa pesi fissi λ_t nella loss (134), calibrati manualmente. Un'alternativa promettente è l'ottimizzazione automatica dei pesi tramite *uncertainty weighting* (Kendall et al., 2018):

$$\lambda_t^{-1} \propto \sigma_t^2, \quad \mathcal{L}_{\text{unc}} = \sum_t \frac{\mathcal{L}_t}{\sigma_t^2} + \log \sigma_t, \tag{141}$$

dove σ_t è la varianza di rumore predetta per il task t, apprendibile come parametro.

4.9 §493 Overfitting e regolarizzazione nel dominio genomico

I dati genomici pongono un problema di *correlazione strutturale*: posizioni genomiche vicine sono altamente correlate (per linkage disequilibrium e struttura chromatiniana). Questo viola l'ipotesi i.i.d. standard e aumenta il rischio di overfitting.

4.9.1 Augmentation specifica al DNA

AlphaGenome usa due strategie di augmentation:

1. **Random shift**: traslazione casuale della finestra genomica di $\pm\delta$ basi, con $\delta \leq 500$ bp, durante il training;
2. **Reverse complement**: con probabilità $1/2$, l'input è sostituito dal suo complemento inverso $\varphi(x)$. Questo è equivalente all'equivarianza Watson-Crick (128) a livello di data augmentation invece di architettura.

4.9.2 Validazione su contigs genomici separati

Una pratica fondamentale nell'AI genomica è la validazione su *contigs* cromosomici non visti in training. La correlazione tra regioni dello stesso cromosoma ha una scala di ~ 1 Mbp (struttura TAD); separare i cromosomi 1-21 per il training e validare su chr22, chrX, chrY garantisce che il modello non stia memorizzando pattern locali.

Nota. Questa pratica è l'equivalente genomico della validazione temporale nei LLM (addestrare su testo prima di una data e validare su testo più recente): garantisce generalizzazione a regioni genomiche strutturalmente diverse da quelle viste in training.

4.10 §494b Attenzione dilatata e gerarchica per sequenze genomiche

La finestra di contesto di 10^6 bp in AlphaGenome richiede meccanismi di attenzione specializzati per evitare il costo $O(N^2)$ con $N \approx 8000$ (dopo il conv-stem). L'*attenzione dilatata* (dilated attention) fornisce un'alternativa.

4.10.1 Attenzione a finestre locali con dilatazione

L'attenzione dilatata di passo d alle posizione i considera solo i token a distanza $\{i-d, i-2d, \ldots, i-Wd, i+d, \ldots, i+Wd\}$, dove W è la larghezza della finestra:

$$A_{ij}^{(\text{dil},d)} = \text{softmax}_j\left(\frac{\langle Q_i, K_j\rangle}{\sqrt{d_k}}\right), \qquad j \in \{i+kd : |k| \le W\}. \tag{142}$$

Con L livelli di dilatazione $d = 1, 2, 4, \ldots, 2^{L-1}$ (come in WaveNet), il campo recettivo effettivo è $O(2^L \cdot W)$ con costo $O(N \cdot W)$ per livello.

Tabella 15: Campo recettivo per diverse strategie di attenzione

Strategia	Costo per layer	Campo recettivo	Esempio
Full attention	$O(N^2)$	N	Transformer standard
Window attention	$O(N \cdot W)$	W	Longformer (locale)
Dilated attention	$O(N \cdot W)$	$2^L \cdot W$	Longformer (globale)
Cross-scale (AlphaGenome)	$O(N_1 N_2)$	$N_1/N_2 \cdot N_2 = N_1$	AlphaGenome

4.10.2 Confronto con l'attenzione cross-scala di AlphaGenome

Mentre l'attenzione dilatata opera a risoluzione singola con salti variabili, AlphaGenome usa risoluzioni *diverse* con comunicazione cross-scala esplicita (125). La differenza chiave è:

Proposizione 4.2 (Informazione persa nell'attenzione dilatata)**.** *L'attenzione dilatata con passo d tratta tutti i token come equidistanti (stessa dimensione di embedding), perdendo la distinzione tra fenomeni che accadono a scale diverse (1 bp vs 1 kbp). AlphaGenome, mantenendo rappresentazioni separate a 16 bp, 128 bp, 1 kbp, 8 kbp, preserva questa distinzione e permette aggiornamenti specifici per scala.*

4.11 §495b Metilazione del DNA e segnali epigenetici come feature aggiuntive

La funzione genomica non è determinata solo dalla sequenza di DNA, ma anche dallo stato epigenetico: metilazione del DNA, acetilazione degli istoni, struttura della cromatina. AlphaGenome predice parte di questi segnali come output, ma le versioni future dei modelli genomici dovranno anche integrarli come input.

4.11.1 La metilazione del DNA

La metilazione dei dinucleotidi CpG (5′-C-G-3′) aggiunge un gruppo metile (CH_3) alla citosina: $C \to 5\text{m}C$. L'encoding one-hot standard $\{A, C, G, T\} \to \mathbb{R}^4$ non distingue C da $5\text{m}C$. L'estensione naturale è un alfabeto di 5 simboli:

$$e_b \in \mathbb{R}^5, \quad b \in \{A, 5\text{m}C, C, G, T\}. \tag{143}$$

4.11.2 Modifiche degli istoni e chromatin accessibility

Lo stato delle modifiche degli istoni (H3K4me3, H3K27ac, H3K27me3, ...) determina l'accessibilità della cromatina, predetta da AlphaGenome (ATAC-seq, §491). Un modello che integri queste informazioni come input necessita di un secondo canale di input "epigenomico" $\mathcal{E} \in \mathbb{R}^{L/r \times M}$, dove M è il numero di modifiche epigenetiche considerate, che può essere fuso con la rappresentazione genomica tramite cross-attention o concatenazione:

$$X^{(\ell)}_{\text{fused}} = \text{MHA}(X^{(\ell)}_{\text{query}},\ \mathcal{E}_{\text{key}},\ \mathcal{E}_{\text{value}}) + X^{(\ell)}. \tag{144}$$

Nota. Questa fusione epigenomica è analoga all'uso dell'MSA in AlphaFold: l'MSA porta informazione evolutiva "esterna" alla sequenza target, e il meccanismo di fusione è matematicamente identico (attenzione cross-modale). La differenza è che l'MSA è trascrizione evolutiva, mentre $\mathcal{E}$ è stato funzionale specifico al tipo cellulare — una forma di *context-conditioning* su variabili biologiche.

4.12 §495c Quantificazione dell'incertezza nelle predizioni genomiche

A differenza di AlphaFold 2, che fornisce un esplicito punteggio di confidenza pLDDT per residuo, AlphaGenome non ha una misura di confidenza per-posizione standard. Questo è un gap metodologico importante.

4.12.1 Conformal Prediction per segnali genomici

La *conformal prediction* (Vovk et al., 2005; Angelopoulos e Bates, 2021) produce intervalli di predizione con garanzie di copertura marginale *senza assunzioni distribuzionali*, usando solo una *calibration set*.

Definizione 4.3 (Intervallo conforme)**.** Dato un modello $\hat{y} = f(x)$ e un calibration set $\mathcal{D}_{\text{cal}}$, il punteggio di non-conformità per il campione i è $s_i = |y_i - \hat{y}_i|$. Per un livello di copertura $1-\alpha$, il quantile:

$$q_\alpha = \lceil (1-\alpha)(1 + 1/|\mathcal{D}_{\text{cal}}|) \rceil\text{-esimo quantile di } \{s_i\}, \tag{145}$$

produce l'intervallo $\hat{C}(x) = [\hat{y}(x) - q_\alpha,\ \hat{y}(x) + q_\alpha]$ che soddisfa $\Pr[y \in \hat{C}(X)] \geq 1-\alpha$ per nuovi campioni i.i.d.

Applicazione ad AlphaGenome. Per la predizione dell'espressione genica (RPKM), il punteggio di non-conformità potrebbe essere $s_i = |\log(\hat{y}_i + 1) - \log(y_i + 1)|$ (errore in spazio log), con il calibration set costituito da regioni genomiche non usate in training. Il quantile (145) produce un intervallo di confidenza su ogni predizione genomica con garanzie distribuzionali-free.

Conformal prediction vs pLDDT: confronto metodologico

Il pLDDT di AlphaFold è una quantità *predetta*: la rete stima quanto sarà accurata la sua predizione. La conformal prediction è una quantità *calibrata esternamente*: usa dati di calibrazione per garantire copertura senza assunzioni sul modello. pLDDT è più granulare (per-residuo) ma richiede training specifico; la conformal prediction è generale ma produce solo intervalli marginali (non per-posizione). Un approccio ibrido — training di una testa pLDDT-genomica calibrata conformal post-hoc — è una direzione di ricerca aperta.

5 Sezione E — Mappa Sistematica delle Differenze Matematiche

5.1 §496 La biforcazione fondamentale: derivazione formale

Tutta la matematica differenziale tra LLM e AI scientifica può essere ricondotta a una singola biforcazione, che deriviamo ora formalmente.

Definizione 5.1 (Spazio di output e gruppo di simmetria naturale)**.** Sia $f_\theta : \mathcal{X} \to \mathcal{Y}$ un modello parametrico. Definiamo il *gruppo di simmetria naturale* di f come il gruppo massimale G^* tale che:

(i) esiste un'azione $\rho_\mathcal{Y}$ di G^* su $\mathcal{Y}$;

(ii) la funzione target f^* (la funzione ideale che f_θ deve approssimare) è G^*-equivariante: $f^*(\rho_\mathcal{X}(g)x) = \rho_\mathcal{Y}(g)f^*(x)$.

Teorema 5.2 (La biforcazione fondamentale)**.** *Le seguenti affermazioni caratterizzano completamente i due tipi di modelli:*

(a) ***LLM****:* $\mathcal{Y} = \{1, \ldots, V\}^T$, $G^* = S_N$ *(gruppo di permutazioni dell'input),* $\mathcal{L} = -\log p_\theta$ *(divergenza KL). La simmetria* S_N *viene rotta dal position encoding.*

(b) ***AlphaFold****:* $\mathcal{Y} = \mathbb{R}^{3N}/\mathrm{SE}(3)$ *(strutture 3D a meno di roto-traslazioni),* $G^* = \mathrm{SE}(3)$ *(roto-traslazioni),* $\mathcal{L} = \mathrm{FAPE}$ *(distanza euclidea in Å). La simmetria* $\mathrm{SE}(3)$ *viene rispettata dall'IPA.*

(c) ***AlphaGenome****:* $\mathcal{Y} = \mathbb{R}^K \times [0,1]^M \times \mathbb{R}^{L \times L}$ *(segnali multi-task),* $G^* = \mathbb{Z}_2$ *(complemento RC),* $\mathcal{L} = \sum_t \lambda_t \mathcal{L}_t$ *(multi-task). La simmetria* $\mathbb{Z}_2$ *viene rispettata dalla media su* $f(x)$ *e* $f(\varphi(x))$*.*

Ogni altra differenza — aggiornamenti triangolari, IPA, FAPE, multi-scala genomica — è una conseguenza *di questa biforcazione, non una scelta arbitraria.*

5.2 §497–500 Tabella comparativa a 15 dimensioni

Tabella 16: Mappa completa LLM vs AlphaFold 2 vs AlphaGenome (15 dimensioni)

Dimensione	**LLM**	**AlphaFold 2**	**AlphaGenome**
Gruppo simmetria	S_N (rotto dal PE)	SE(3) (rispettato da IPA)	$\mathbb{Z}_2$ (WC, test-time aug.)
Tipo input	Seq. discreta 1D	Seq. discreta 1D + MSA 2D	Seq. discreta 1D multi-scala
Struttura features	$(N \times d)$	$(N \times d) + (N \times N \times d)$	4 scale + pair opzionale
Meccanismo chiave	MHA + causal mask	Evoformer + IPA	Conv tower + cross-scale attn
Aggiornamenti triang.	No	Sì (4 varianti, §468)	No (ma head bilineare Hi-C)
Position encoding	RoPE, ALiBi, sin	Discreta (seq) + pairwise	Log-distanza genomica
MSA / evolutivo	No	Sì (asse critico)	No (sequenza diretta)
Obiettivo training	Cross-entropy	FAPE + ausiliari geom.	Multi-task (Pearson, Poisson, ...)

Dimensione	LLM	AlphaFold 2	AlphaGenome
Output	Distribuz. token	Coordinate 3D + confidenza	Profili funzionali + Hi-C
Unità output	bit	Ångström	RPKM / sigmoid / count
Causalità	Sì (causal mask)	No (bidirezionale)	No (bidirezionale)
Recycling	No	Sì (3 iter. std.)	Parziale
Confidenza per token	pLPP (implicita)	pLDDT per residuo	Var. predetta (implicita)
Scaling behaviour	Legge Chinchilla	Non segue stessa legge	Più simile agli LLM
Dim. contesto	10^3–10^6 token	50–2500 residui	$\sim$ 8000 token (dopo stem)

5.3 §501 La rappresentazione doppia: dimostrazione di necessità

Un LLM processa una sequenza $(x_1, \ldots, x_N)$ e mantiene una rappresentazione $h_i \in \mathbb{R}^d$ per ogni token: struttura lineare $O(N)$ nello spazio. AlphaFold mantiene due rappresentazioni:

$$s_i \in \mathbb{R}^{c_s}, \quad i = 1, \ldots, L \quad \text{(sequenza)}, \tag{146}$$

$$z_{ij} \in \mathbb{R}^{c_z}, \quad i, j = 1, \ldots, L \quad \text{(coppia)}, \tag{147}$$

con la rappresentazione di coppia di dimensione $O(L^2)$.

Proposizione 5.3 (Necessità della rappresentazione di coppia)**.** *Nessuna funzione $f : \mathbb{R}^{L \times c_s} \to \mathbb{R}^{3L}$ che operi solo sulla rappresentazione di sequenza può apprendere in modo sample-efficient la struttura 3D di proteine con interazioni a lungo raggio dominate da coppie di residui distanti nella sequenza ma vicini nello spazio 3D.*

Argomentazione. Siano i e j due residui con $|i - j| > k$ per una soglia k. Nella struttura 3D, la distanza fisica d_{ij} può essere piccola (contatto), nonostante la grande distanza sequenziale. Una rete che opera solo sulla sequenza deve trasmettere l'informazione sul contatto (i, j) attraverso una catena di interazioni di residui intermedi, richiedendo profondità $\Omega(|i - j|/r)$ per un campo recettivo r.

Con $L = 500$ e contatti a distanza sequenziale 200, questo richiederebbe $\sim 200/r$ strati — impraticabile. La rappresentazione di coppia z_{ij} codifica *direttamente* la probabilità di contatto tra i e j, indipendentemente dalla distanza sequenziale, rendendo il problema trattabile con profondità costante. □ □

Il costo della rappresentazione di coppia. Ogni aggiornamento dell'Evoformer richiede $O(L^2 c_z)$ memoria e $O(L^2 c_z d_h)$ operazioni. Per $L = 500$, $c_z = 128$:

$$500^2 \times 128 \times 4 \text{ byte} \approx 128 \text{ MB}$$

solo per la matrice di coppia, paragonabile alla memoria di un LLM con miliardi di parametri.

Perché AlphaGenome non usa la rappresentazione di coppia completa. Per $L_0 = 10^6$ basi (prima del downsamplig), la matrice di coppia richiederebbe $(10^6)^2 \times 4$ byte $= 4$ TB — impossibile. AlphaGenome rinuncia alla rappresentazione di coppia completa, compensando con la struttura multi-scala e l'attenzione cross-scala. La testa bilineare Hi-C (131) ricostruisce una matrice di coppia solo per l'output finale, non come rappresentazione interna.

5.4 §502 Confronto delle loss: dalla bit-entropia alla geometria

5.4.1 Cross-entropy (LLM)

$$\mathcal{L}_{\text{LM}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}). \tag{148}$$

Il gradiente rispetto al logit $\ell_k = \log p_k$ per il token vero k^* è:

$$\frac{\partial \mathcal{L}_{\text{LM}}}{\partial \ell_k} = p_k - \mathbf{1}[k = k^*]. \tag{149}$$

Geometricamente, è la distanza KL tra la distribuzione vera (one-hot) e quella predetta (softmax): non c'è distanza euclidea, non c'è geometria fisica.

5.4.2 FAPE (AlphaFold 2)

$$\frac{\partial \mathcal{L}_{\text{FAPE}}}{\partial \hat{x}_a} = \frac{1}{LN_{\text{atoms}}} \sum_i \frac{T_i^{-\top} T_i^{-1}(\hat{x}_a - x_a)}{\|T_i^{-1}(\hat{x}_a - x_a)\| + \varepsilon}. \tag{150}$$

È la somma di versori unitari (in coordinate locali) puntanti dall'atomo predetto verso quello vero. È un segnale geometrico nel senso più letterale.

5.4.3 Pearson correlation loss (AlphaGenome)

$$\frac{\partial \mathcal{L}_{\text{Pearson}}}{\partial \hat{y}_i} = -\frac{1}{\sigma_{\hat{y}}}\left[\frac{y_i - \bar{y}}{\sigma_y} - r_{\text{Pearson}} \cdot \frac{\hat{y}_i - \bar{\hat{y}}}{\sigma_{\hat{y}}}\right], \tag{151}$$

dove $r_{\text{Pearson}} = \text{Cov}(\hat{y}, y)/(\sigma_{\hat{y}}\sigma_y)$ è la correlazione corrente. Il gradiente spinge la distribuzione predetta a correlarsi con quella vera, senza penalizzare differenze di scala.

Tabella 17: Confronto sistematico delle funzioni di perdita

Proprietà	**Cross-entropy**	**FAPE**	**Pearson / Poisson**
Spazio output	Probabilità (simplesso)	$\mathbb{R}^3$ (Ångström)	$\mathbb{R}$ (RPKM, conteggi)
Unità	bit	Å	dimensionless / count
Geometria	No (KL divergence)	Sì (euclidea)	No (statistica)
Invarianza	Permutazione token	SE(3)	Scala / offset (Pearson)
Gradiente	$p_k - \delta_{kk^*}$	Versori geometrici	Basato su correlazione
Outlier	Gestiti da softmax	Gestiti da clamping	Gestiti da log (Poisson)
Task	Classificazione token	Regressione 3D	Regressione biologica

5.5 §503 pLDDT vs confidenza nei LLM

5.5.1 pLDDT in AlphaFold 2

AlphaFold 2 fornisce per ogni residuo i un punteggio di confidenza $\text{pLDDT}_i \in [0, 100]$, predetto come classificatore di 50 bin del valore LDDT (Local Distance Difference Test):

$$\text{LDDT}_i = \frac{1}{|N_i|}\sum_{j \in N_i} \frac{1}{4} \sum_{\tau \in \{0.5,1,2,4\}} \mathbf{1}\Big[|d_{ij} - \hat{d}_{ij}| < \tau\Big]. \tag{152}$$

Il pLDDT è la probabilità predetta che l'LDDT del residuo i sia alto.

5.5.2 Confidenza nei LLM: token probability e perplexity

La confidenza di un LLM si manifesta come la distribuzione $p_\theta(x_t \mid x_{<t})$: un token con alta probabilità assegnata è "sicuro". La perplexity è:

$$\mathrm{PPL} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{<t})\right) = \exp(\mathcal{L}_{\mathrm{LM}}). \tag{153}$$

Tabella 18: Confronto delle misure di confidenza

Proprietà	**pLDDT (AlphaFold)**	**Token probability (LLM)**
Tipo	Predizione esplicita	Distribuzione implicita
Scala	$[0, 100]$ (LDDT)	$[0, 1]$ (probabilità)
Cosa misura	Correttezza strutturale fisica	Probabilità del token successivo
Unità	Distanza (Å, indiretta)	Adimensionale
Calibrazione	Rispetto a strutture vere	Rispetto alla frequenza del corpus
Uso a test time	Sì (filtraggio regioni)	Sì (beam search, sampling)

5.6 §504 Recycling vs Chain-of-Thought: due forme di iterazione

Il recycling di AlphaFold (92) e il chain-of-thought dei LLM (Vol. II, §65) sono entrambi meccanismi di *iterazione* che migliorano la qualità dell'output. Ma le loro matematiche sono profondamente diverse.

Recycling (AlphaFold).

- La struttura predetta $\{T_i^{(t)}\}$ viene reinserita come feature all'inizio del passo $t+1$;
- Il gradiente fluisce attraverso i passi di recycling durante il training;
- L'obiettivo è la convergenza a un punto fisso: $\{T_i^{(t+1)}\} \approx \{T_i^{(t)}\}$;
- Il miglioramento è geometrico: la struttura diventa sempre più coerente con se stessa.

Chain-of-thought (LLM).

- I token intermedi ("pensieri") vengono generati autoregressivamente;
- Il modello li usa come contesto per generare la risposta finale;
- Non c'è retroalimentazione strutturale: i token intermedi non modificano le rappresentazioni interne, solo il contesto;
- Il miglioramento è semantico: il ragionamento esplicito guida la risposta.

Osservazione 5.4. Un analogo più preciso del recycling nei LLM sarebbe l'*iterative refinement*: generare una risposta iniziale, poi chiudere il prompt con quella risposta e rigenerare. Questo meccanismo è usato in alcuni sistemi di reasoning avanzati (o1, DeepSeek-R1), ma non è parte del forward pass standard di un LLM.

Tabella 19: Recycling vs Chain-of-Thought: confronto matematico

Proprietà	**Recycling (AF)**	**CoT (LLM)**
Tipo di iterazione	Punto fisso geometrico	Generazione autoregressiva
Feedback	Strutturale (frames T_i)	Semantico (token nel contesto)
Gradiente durante training	Sì (backprop attraverso iter.)	Sì (RLHF / supervised)
Numero di iterazioni	Fisso (tipicamente 3)	Variabile (dipende dal task)
Output intermedio	Non visibile all'esterno	Visibile (testo prodotto)
Convergenza	Verso punto fisso	Non garantita

5.7 §500a Implicazioni computazionali della biforcazione: analisi della complessità

La biforcazione tra S_N, SE(3) e $\mathbb{Z}_2$ come gruppi di simmetria si traduce direttamente in differenze di complessità computazionale.

5.7.1 Complessità dell'inferenza

Tabella 20: Complessità computazionale a confronto

Operazione	**LLM**	**AlphaFold 2**	**AlphaGenome**
Self-attention	$O(N^2 d)$	$O(L^2 c_s)$	$O(N_\ell^2 d)$ per ℓ
Rappresentazione coppia	N/A	$O(L^2 c_z)$ mem.	Parziale (Hi-C out.)
Aggiornamenti triangolari	N/A	$O(L^3 c_z)$ FLOPs	N/A
IPA	N/A	$O(L^2 HK)$	N/A
Cross-scale attention	N/A	N/A	$O(N_1 N_2 d)$
KV-cache (autoregress.)	$O(N \cdot d_k \cdot 2H)$	N/A	N/A

Il collo di bottiglia dell'Evoformer. L'aggiornamento triangolare moltiplicativo ha complessità $O(L^3)$: per $L = 500$, questo è 125×10^6 operazioni solo per questo componente. È per questo che AlphaFold 3 introduce il Pairformer che riduce il numero di aggiornamenti triangolari e usa Flash Attention sulla rappresentazione di coppia.

5.7.2 Analisi della finestra di contesto effettiva

Definizione 5.5 (Campo recettivo effettivo)**.** Il campo recettivo effettivo r_{eff} di un modello è la distanza massima (in unità di token) entro cui un token può influenzare la predizione di un altro token.

- **LLM**: $r_{\text{eff}} = N$ (causal mask limita il futuro, ma non il passato). Con RoPE e ALiBi si estende fino alla finestra di contesto $N_{\max}$.

- **AlphaFold**: $r_{\text{eff}} = L$ per la rappresentazione di sequenza, e $r_{\text{eff}} = L$ anche per la coppia (l'Evoformer è bidirezionale completo). Il recycling estende ulteriormente le interazioni attraverso la struttura predetta.
- **AlphaGenome**: $r_{\text{eff}} = 1\,\text{Mbp}$ in basi, ma limitato dalla gerarchia dei livelli: a livello fine (16 bp), $r_{\text{eff}} \approx 128\,\text{kbp}$ (dovuto al context window a quella risoluzione); a livello grossolano (8 kbp), $r_{\text{eff}} = 1\,\text{Mbp}$.

5.8 §500b Analisi della generalization: teoria PAC

La teoria PAC-Bayes offre un framework per confrontare la generalizzazione dei diversi modelli in termini delle loro simmetrie.

Teorema 5.6 (Bound di generalizzazione PAC-Bayes con simmetria)**.** *Sia G il gruppo di simmetria di un modello f e sia $\mu(G)$ la dimensione (numero di elementi per G finito, dimensione di Haar per G continuo). Il bound di generalizzazione McAllester–Seeger modificato per simmetria è:*

$$\mathcal{L}_{true} \leq \hat{\mathcal{L}}_{train} + \sqrt{\frac{\mathrm{KL}(p\|q) + \log(1/\delta)}{2m} - \frac{\log \mu(G)}{m}}, \tag{154}$$

dove m è la dimensione del dataset e il termine $-\log \mu(G)/m$ riflette la riduzione della complessità del modello grazie alla simmetria.

Per $G = S_N$ (LLM senza PE): $\mu(S_N) = N!$, quindi $\log \mu(G) = O(N \log N)$ — il modello è fortemente regolarizzato dalla simmetria di permutazione. Per $G = \mathrm{SE}(3)$ (AlphaFold): $\mu(G) = \infty$ (gruppo continuo), ma il termine di regolarizzazione è proporzionale alla misura di Haar integrale sul gruppo compatto $\mathrm{SO}(3)$, che è finita e grande — regolarizzazione ancora maggiore.

5.9 §500c Complessità di Rademacher e generalizzazione comparata

Il bound di generalizzazione PAC-Bayes (154) usa la simmetria come proxy per la complessità del modello. Complementiamo questa analisi con la complessità di Rademacher, che offre bounds più direttamente calcolabili per architetture specifiche.

Definizione 5.7 (Complessità di Rademacher empirica)**.** Sia $\mathcal{H}$ una classe di ipotesi, $S = \{x_i\}_{i=1}^m$ un campione. La complessità di Rademacher empirica è:

$$\hat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i h(x_i) \right], \tag{155}$$

dove $\sigma_i \sim \mathrm{Uniform}(\{-1, +1\})$ sono variabili di Rademacher (rumore casuale simmetrico).

Teorema 5.8 (Bound di generalizzazione via Rademacher)**.** *Per qualsiasi $\delta > 0$, con probabilità almeno $1 - \delta$ su un campione di m esempi, per ogni $h \in \mathcal{H}$:*

$$L(h) \leq \hat{L}(h) + 2\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}. \tag{156}$$

5.9.1 Stima della complessità di Rademacher per Transformer

Per un Transformer con pesi $\|W\|_F \leq B$ e attivazioni Lipschitz-1, la complessità cresce come:

$$\mathfrak{R}_m(\mathcal{H}_{\text{Transf}}) \leq O\left(\frac{B^{2L} \sqrt{L \log N}}{m} \right), \tag{157}$$

dove L è il numero di layer e N la lunghezza della sequenza. Il termine $\sqrt{L \log N}$ riflette il costo dell'attenzione: ogni layer aumenta logaritmicamente la complessità in N a causa del softmax.

5.9.2 Implicazione per AlphaFold

Per AlphaFold 2, la rappresentazione di coppia $z_{ij} \in \mathbb{R}^{L^2 \times c_z}$ ha dimensione $O(L^2)$. La norma Frobenius dei parametri degli aggiornamenti triangolari cresce come $O(L^2)$ nel caso peggiore, aumentando $\mathfrak{R}_m$. Tuttavia, i vincoli fisici (distanze positive, disuguaglianza triangolare) limitano lo spazio delle soluzioni ammissibili, riducendo il gap tra la complessità nominale e quella effettiva — motivando nuovamente l'impatto degli inductive bias fisici.

5.10 §503b Calibrazione dei modelli: temperature scaling, Platt, isotonica

Un modello è *calibrato* se la sua confidenza $p(y = k \mid x)$ corrisponde alla frequenza empirica di correttezza: su tutti gli input dove il modello assegna probabilità 0.8 alla classe k, circa l'80% delle volte la classe giusta è effettivamente k.

5.10.1 Expected Calibration Error (ECE)

Definizione 5.9 (ECE)**.** Partizioniamo i campioni in B bin $\{M_b\}_{b=1}^{B}$ basati sulla confidence $\hat{p}$. Sia $\mathrm{acc}(M_b)$ l'accuratezza e $\mathrm{conf}(M_b)$ la confidenza media nel bin b. L'ECE è:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{|M_b|}{m} |\mathrm{acc}(M_b) - \mathrm{conf}(M_b)|. \tag{158}$$

5.10.2 Temperature Scaling

Il metodo più semplice di post-hoc calibration è il *temperature scaling*: si sostituisce $\hat{p}(y \mid x) = \mathrm{softmax}(\ell(x)/T)$, dove $T > 0$ è una temperatura scalare ottimizzata su un validation set:

$$T^* = \arg\min_{T} \mathcal{L}_{\mathrm{NLL}}(\{(\ell_i, y_i)\}, T). \tag{159}$$

Per $T > 1$ le probabilità vengono appiattite (modello sotto-confidente diventa meglio calibrato); per $T < 1$ vengono accentuate.

Applicazione ad AlphaFold. Il pLDDT di AlphaFold è già ottimizzato durante il training come un classificatore a 50 bin. Non richiede temperature scaling a posteriori, poiché la sua loss di training (cross-entropy sull'LDDT binned) è già calibrante per costruzione. La calibrazione del pLDDT è stata verificata empiricamente su CASP14: le regioni con pLDDT > 90 hanno RMSD medio < 1 Å rispetto alle strutture cristallografiche.

Applicazione ai LLM. I LLM moderni mostrano sistematicamente *over-confidence*: la probabilità assegnata al token selezionato è spesso troppo alta. Temperature scaling con $T \approx 1.2$–2 è un approccio standard per migliorare la calibrazione dei token-level logit. Per la calibrazione di output più lunghi (es. "quanto è sicuro questo fatto"), sono necessarie tecniche più elaborate come *verbalized uncertainty* ("Sono sicuro al 70% che...") o *conformal sets* a livello di sequenza.

5.11 §504b Prospettiva dell'informazione teorica: LLM vs AI scientifica

La teoria dell'informazione offre una prospettiva unificante sulle differenze tra i modelli.

5.11.1 Minimum Description Length e architettura

Il principio MDL (Rissanen, 1978) afferma che il miglior modello di dati $\mathcal{D}$ è quello che minimizza la lunghezza di codifica congiunta $|\mathcal{D}| + |M|$:

$$M^* = \arg\min_M \left[-\log p(D \mid M) + \text{desc}(M) \right], \tag{160}$$

dove $\text{desc}(M)$ è il costo di descrizione del modello M.

Per un LLM, $-\log p(\mathcal{D} \mid M)$ è la perplexity (in bit) sul corpus, e $\text{desc}(M)$ è proporzionale al numero di parametri (costo di codifica dei pesi). L'ottimizzazione di N e D nel framework Chinchilla (22) è un'istanza del principio MDL.

Per AlphaFold, la situazione è fondamentalmente diversa: i dati strutturali (coordinate 3D) hanno una dimensione intrinseca molto più bassa del corpus linguistico (le strutture proteiche vivono su una varietà di dimensione $O(L)$ in $\mathbb{R}^{3L}$, mentre il testo non ha una varietà a dimensione finita). Il costo descrittivo MDL di una struttura proteica è:

$$\text{desc}(\text{struttura}) \approx L \cdot \log \frac{1}{\epsilon_{\text{RMSD}}}, \tag{161}$$

dove $\epsilon_{\text{RMSD}} \sim 1$ Å è la risoluzione di interesse. Per $L = 500$ e $\epsilon = 1$ Å, questo è $\sim 500 \cdot 3$ bit (usando la discretizzazione di spazio 3D a 1 Å): molto meno dei miliardi di bit per descrivere un corpus linguistico.

5.11.2 Mutua informazione tra sequenza e struttura/funzione

L'obiettivo fondamentale di AlphaFold e AlphaGenome è massimizzare la mutua informazione tra l'input (x: sequenza) e l'output (y: struttura o funzione):

$$I(X;Y) = H(Y) - H(Y \mid X), \tag{162}$$

dove H è l'entropia di Shannon. Per le proteine, la mutua informazione $I(\text{sequenza}; \text{struttura})$ è enorme: la struttura è quasi completamente determinata dalla sequenza (dogma di Anfinsen). Per il linguaggio, $I(\text{sequenza precedente}; \text{token successivo})$ è molto più bassa: il testo ha alta entropia condizionale.

Tabella 21: Prospettiva informazionale comparata

Proprietà	**LLM**	**AlphaFold**	**AlphaGenome**
$H(Y)$ output	Alta (vocabolario 10^5)	Media (strutture 3D)	Media (segnali biologici)
$H(Y \mid X)$ cond.	Alta	Bassa (Anfinsen)	Media
$I(X;Y)$ mut. info	Media	Alta	Alta
Dim. intrinseca output	∞ (testo)	$O(L)$ ($\mathbb{R}^3$ vincolato)	$O(L)$ (segnali 1D)
Distorsione adm.	0 (token esatti)	~ 1 Å (RMSD)	$r > 0.8$ (Pearson)

6 Sezione F — Convergenza Futura: Scientific Foundation Models

6.1 §505 Il continuum LLM–AI scientifica

La distinzione tra LLM e AI scientifica non è un'opposizione binaria ma un *continuum* parametrizzato dal grado di vincolo fisico incorporato nell'architettura. Definiamo formalmente:

Definizione 6.1 (Indice di vincolo fisico)**.**

$$\mathcal{I}(f) = \frac{\text{Informazione geometrica incorporata}}{\text{Flessibilità appresa dai dati}}. \tag{163}$$

I modelli si collocano lungo questo continuum:

GPT-4	LLaMA	ESM-2	AlphaGenome	ESM-3	AlphaFold 2	DFT
↓	↓	↓	↓	↓	↓	↓
			$\mathcal{I}$ crescente $\longrightarrow$			
nessuna geometria	nessuna	implicita evoluzione	$\mathbb{Z}_2$	VQ-SE(3)	SE(3) piena	ab initio

Implicazione per il design. Maggiore è $\mathcal{I}$, minore è il numero di parametri necessari per raggiungere una data performance sul dominio specifico — ma minore è anche la generalizzabilità a domini diversi da quello per cui il vincolo è stato costruito. Il design di un Scientific Foundation Model è essenzialmente l'ottimizzazione di questo trade-off.

6.2 §506 ESM-3: convergenza in atto

ESM-3 (Hayes et al., *Science* 2024) è il primo modello che è simultaneamente un Protein Language Model (PLM) e un predittore strutturale, realizzando la convergenza tra LLM e AI scientifica.

6.2.1 Architettura di ESM-3

ESM-3 usa un obiettivo di *masked token prediction* — identico a BERT — ma su tre modalità condizionate l'una sull'altra:

$$\mathcal{L}_{\text{ESM-3}} = \mathcal{L}_{\text{seq}} + \mathcal{L}_{\text{struct}} + \mathcal{L}_{\text{func}}, \tag{164}$$

dove:

- $\mathcal{L}_{\text{seq}}$: cross-entropy sui token amminoacidici (come BERT);
- $\mathcal{L}_{\text{struct}}$: cross-entropy sui *token strutturali* (la struttura è quantizzata in un VQ-VAE);
- $\mathcal{L}_{\text{func}}$: cross-entropy sui token funzionali (annotazioni Gene Ontology).

6.2.2 Quantizzazione strutturale con VQ-VAE

Il VQ-VAE (Vol. II, §82) converte le coordinate 3D continue in token discreti. L'encoder apprende una mappa $\text{Enc} : \mathbb{R}^{3L} \to \mathbb{R}^{L \times d_z}$ e il codebook è $\mathcal{C} = \{e_k\}_{k=1}^{K} \subset \mathbb{R}^{d_z}$. La quantizzazione è:

$$z_i^q = e_{k^*}, \quad k^* = \arg\min_k \|z_i - e_k\|^2, \tag{165}$$

con perdita:

$$\mathcal{L}_{\text{VQ}} = \underbrace{\|\text{sg}[z_i] - e_{k^*}\|^2}_{\text{codebook}} + \beta \underbrace{\|z_i - \text{sg}[e_{k^*}]\|^2}_{\text{commitment}}, \tag{166}$$

dove $\text{sg}[\cdot]$ è il *stop-gradient*. La struttura è così rappresentata come una sequenza di token discreti: il problema strutturale diventa formalmente identico al problema linguistico.

Il trade-off della quantizzazione. La discretizzazione perde precisione geometrica: con $K = 4096$ centroidi, la risoluzione è ~ 0.5 Å per residuo — accettabile per molti task (classificazione funzionale, design di proteine) ma insufficiente per la cristallografia ad alta precisione. AlphaFold 2, operando con coordinate continue, non ha questa limitazione.

6.3 §507 Scientific Foundation Models: architettura unificata

Le architetture SFM emergenti combinano matematicamente quattro componenti:

1. **Backbone Transformer condiviso**, con context window estesa e Flash Attention;
2. **Encoding di simmetria specializzato** per ogni modalità:
   - RoPE per sequenze testuali (S_N-equivariante);
   - IPA per strutture proteiche (SE(3)-invariante);
   - Log-PE per sequenze genomiche ($\mathbb{Z}_2$-equivariante);
   - Armoniche sferiche per molecole piccole ($O(3)$-equivariante);
3. **Teste di decodifica multiple**: autoregressiva per testo, FAPE o diffusione per struttura, Poisson/Pearson per genomica;
4. **Cross-modal attention**: permette al modello di condizionare la predizione proteica su dati genomici e viceversa.

Modelli recenti verso SFM.

Evo-2 (Arc Institute, 2025) : modello a 40 miliardi di parametri addestrato su 9.3×10^{12} nucleotidi di sequenze biologiche di tutte le specie. Usa un'architettura ibrida Transformer + SSM (Mamba), con una finestra di contesto di 1 Mbp. Genera sequenze genomiche, predice varianti patogene, e modella l'evoluzione genomica.

Chai-1 (Chai Discovery, 2024) : modello per la struttura di complessi biomolecolari (proteine, DNA, RNA, ligandi) basato su diffusione, competitivo con AlphaFold 3 senza MSA per molti target.

Boltz-1 (MIT, 2025) : alternativa open-source ad AlphaFold 3 per strutture molecolari generali, con architettura basata su diffusione e Pairformer.

6.4 §508 Il Neural Scaling per l'AI scientifica

Una domanda aperta fondamentale è: le leggi di scala (16)–(22) valgono per l'AI scientifica?

6.4.1 Scaling dei protein language model

Per i PLM come ESM-2, le prestazioni su benchmark strutturali seguono approssimativamente una power law nel numero di parametri, con esponente simile a quello dei LLM per $N < 10^{10}$:

$$\text{TM-score}(N) \approx 1 - \left(\frac{N_c'}{N}\right)^{\alpha'}, \quad \alpha' \approx 0.05\text{–}0.10. \tag{167}$$

Tuttavia, AlphaFold 2 ha solo ~ 93 milioni di parametri e supera ESM-2 da 15 miliardi di parametri sul task strutturale. La ragione è l'inductive bias geometrico (IPA, aggiornamenti triangolari, FAPE), che vale matematicamente come una quantità enorme di training data strutturale aggiuntivo.

6.4.2 Scaling laws con inductive bias: teoria

Definizione 6.2 (Informazione geometrica incorporata $\mathcal{I}(f)$)**.** Definiamo l'inductive bias fisico di un modello f come la quantità di informazione geometrica che riduce il numero effettivo di esempi necessari per raggiungere una data perdita.

Proposizione 6.3 (Legge di scala modificata)**.** *Se $\mathcal{I}(f)$ denota l'inductive bias fisico di un modello f, una legge di scala modificata è:*

$$L(N, \mathcal{I}) \approx \left(\frac{N_c(\mathcal{I})}{N}\right)^{\alpha}, \quad N_c(\mathcal{I}) \text{ decrescente in } \mathcal{I}. \tag{168}$$

Maggiore è l'inductive bias fisico, minore è il numero critico N_c, cioè minore è il numero di parametri necessario per raggiungere una data performance.

Implicazione pratica. Il design ottimale di un SFM è il problema di massimizzare $\mathcal{I}(f)$ mantenendo la generalizzabilità del modello a nuovi domini. Questo è equivalente a trovare il gruppo di simmetria massimale che vale *su tutti* i domini di interesse: le simmetrie condivise (invarianza alle permutazioni dell'ordine dei campioni del training set, invarianza alla scala dell'output in assenza di calibrazione assoluta) vengono incorporate, quelle dominio-specifiche (SE(3) per strutture, $\mathbb{Z}_2$ per genoma) vengono usate solo per le componenti di output corrispondenti.

6.5 §509 La matematica dell'emergenza nei Scientific Foundation Models

Nei LLM, le abilità emergenti (Vol. IV, §§219–224) si manifestano come discontinuità nelle curve di scaling: capacità che appaiono improvvisamente oltre una soglia di parametri. Per i SFM scientifici il fenomeno analogo è l'*emergenza cross-modale*: la capacità di un modello addestrato su sequenze proteiche di predire strutture genomiche (o viceversa) senza training esplicito su quel task.

6.5.1 L'ipotesi dello spazio biologico universale

L'ipotesi alla base è che le rappresentazioni latenti di sequenze biologiche diverse (amminoacidiche e nucleotidiche) condividano uno spazio geometrico sottostante connesso dall'evoluzione:

$$h^{(\text{prot})} \approx \varphi(h^{(\text{DNA})}) \quad \text{per sequenze biologicamente correlate,} \tag{169}$$

dove φ è una funzione di allineamento apprendibile tra i due spazi di embedding.

Proposizione 6.4 (Condizione di emergenza cross-modale)**.** *L'emergenza cross-modale avviene quando la funzione di allineamento φ in (169) è approssimabile con un numero di parametri piccolo rispetto al modello totale, il che accade se e solo se le due modalità condividono una struttura latente comune di dimensione effettiva $d_{\text{eff}} \ll d$.*

6.5.2 Evidenza empirica

- ESM-3, addestrato principalmente su sequenze proteiche, riesce a generare proteine funzionali condizionate solo su annotazioni funzionali Gene Ontology — un form di emergenza cross-modale (struttura → funzione).
- Evo-2, addestrato su sequenze genomiche, predice accuratamente l'essenzialità dei geni e l'effetto di varianti rare — task mai visti esplicitamente nel training.
- AlphaGenome, che predice contemporaneamente espressione genica e struttura 3D del cromosoma (Hi-C), mostra trasferimento implicito tra i due task: migliorare la rappresentazione per Hi-C migliora l'espressione genica senza supervisione esplicita.

6.6 §510 Sintesi finale e prospettive

Teorema di struttura (informale)

Sia $\mathcal{F}$ la famiglia di tutti i modelli basati su Transformer con training gradient-based. La partizione di $\mathcal{F}$ in LLM e AI scientifica è determinata da:

(1) Il gruppo di simmetria G del dominio target (S_N, SE(3), $\mathbb{Z}_2$, ...);

(2) La metrica naturale sull'output (divergenza KL, norma euclidea, Pearson, Poisson, ...);

(3) La struttura delle features imposta da G (1D, 2D coppia, multi-scala, ...).

Il substrato computazionale (attenzione, layer norm, Adam, scaling) è condiviso e indipendente da G.

Le tre grandi direzioni di ricerca che emergono da questa analisi sono:

1. Architetture universalmente equivarianti. Reti che si adattano automaticamente al gruppo di simmetria del dominio, senza richiedere un design specifico per dominio. Il candidato più promettente è la famiglia delle *geometric algebra networks*, che rappresenta tutte le caratteristiche come multivettori nell'algebra di Clifford $\mathcal{G}(3,0,1)$ (l'algebra di Lie di SE(3)) e applica prodotti geometrici come operazioni fondamentali.

2. Leggi di scala con inductive bias. Formalizzazione rigorosa di come l'inductive bias fisico modifica le leggi di Kaplan/Chinchilla, con implicazioni per il design ottimale di SFM. La proposta è di definire un *numero effettivo di parametri*:

$$N_{\text{eff}}(f) = N_{\text{params}}(f) \cdot \exp(\mathcal{I}(f)), \tag{170}$$

tale che modelli con diverso $\mathcal{I}$ ma identico N_{eff} abbiano prestazioni comparabili.

3. Loss geometriche differenziabili unificatrici. Lo sviluppo di funzioni di costo che interpolino continuamente tra la cross-entropy linguistica e la FAPE geometrica, abilitando training congiunto su domini eterogenei:

$$\mathcal{L}_{\text{unified}}(\alpha) = (1-\alpha)\,\mathcal{L}_{\text{CE}} + \alpha\,\mathcal{L}_{\text{FAPE}}, \quad \alpha \in [0,1], \tag{171}$$

dove α è un parametro di bilanciamento adattivo che dipende dal tipo di output richiesto. La sfida è che le unità sono incompatibili (bit vs Å): una possibile normalizzazione è $\mathcal{L}_{\text{CE}}/H_{\max}$ (bit normalizzati all'entropia massima) e $\mathcal{L}_{\text{FAPE}}/\varepsilon_c$ (Å normalizzati alla soglia di clamping).

Il messaggio finale. La matematica che accomuna LLM e AI scientifica è reale, profonda, e computazionalmente fondamentale. La matematica che li distingue non è meno reale: è la geometria del mondo fisico che si impone sulla struttura del modello. Comprendere questa imposizione — formalizzarla, misurarla, sfruttarla — è il passo necessario per costruire la prossima generazione di modelli scientifici universali: modelli che leggano il genoma, predicano la struttura delle proteine, progettino farmaci, e sintetizzino letteratura scientifica, tutto all'interno di un unico framework matematico coerente.

6.7 §506b State Space Models e Mamba per sequenze biologiche lunghissime

Evo-2 (Arc Institute, 2025), citato in §507, usa un'architettura ibrida Transformer + State Space Model (SSM/Mamba) per modellare sequenze genomiche di 1 Mbp. Deriviamo la matematica degli SSM.

6.7.1 SSM lineari e la connessione con i sistemi dinamici

Un *Linear State Space Model* discreto è definito da:

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \tag{172}$$
$$y_t = Ch_t, \tag{173}$$

dove $h_t \in \mathbb{R}^N$ è lo stato nascosto, $x_t \in \mathbb{R}$, $y_t \in \mathbb{R}$, $\bar{A} \in \mathbb{R}^{N\times N}$, $\bar{B} \in \mathbb{R}^{N\times 1}$, $C \in \mathbb{R}^{1\times N}$. La struttura $(\bar{A}, \bar{B}, C)$ è tipicamente parametrizzata in forma diagonale $\bar{A} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ per efficienza computazionale.

Discretizzazione dal tempo continuo. L'SSM continuo $h'(t) = Ah(t) + Bx(t)$ viene discretizzato con passo Δ:

$$\bar{A} = e^{\Delta A}, \quad \bar{B} = (\Delta A)^{-1}(e^{\Delta A} - I)\Delta B. \tag{174}$$

Il parametro $\Delta > 0$ controlla la "granularità temporale" dell'SSM: Δ grande corrisponde a integrare su intervalli lunghi, Δ piccolo a una risposta locale.

6.7.2 Mamba: SSM selettivo con gating

La limitazione degli SSM lineari classici è che $(\bar{A}, \bar{B}, C)$ non dipendono dall'input: il modello non può "selezionare" cosa ricordare o dimenticare in funzione del contesto. Mamba (Gu e Dao, 2023) risolve questo rendendo Δ, B, C funzioni dell'input:

$$\Delta_t = \mathrm{softplus}(\mathrm{Linear}(x_t)) > 0, \tag{175}$$
$$B_t = \mathrm{Linear}(x_t) \in \mathbb{R}^N, \tag{176}$$
$$C_t = \mathrm{Linear}(x_t) \in \mathbb{R}^N, \tag{177}$$
$$h_t = \bar{A}(\Delta_t)h_{t-1} + \bar{B}(\Delta_t, B_t)x_t. \tag{178}$$

Proposizione 6.5 (SSM selettivo come attenzione lineare)**.** *Il Mamba selettivo è equivalente a un meccanismo di attenzione con kernel $k(t,s) = C_t^\top e^{\bar{A}(t-s)}B_s$ per $s \le t$. La selezione tramite Δ_t è analoga alla causal mask nell'attenzione dei LLM, ma implementata come memoria esponenzialmente decadente invece di prodotto scalare.*

6.7.3 Confronto computazionale: Mamba vs Transformer

Tabella 22: Confronto computazionale Mamba vs Transformer per sequenze lunghe

Operazione	**Transformer**	**Mamba**	**Evo-2 (ibrido)**
Training	$O(N^2 d)$	$O(Nd)$	$O(N^{1.5}d)$ (approx)
Inferenza (passo)	$O(Nd)$ (KV-cache)	$O(d)$ (stato fisso!)	$O(d)$
Memoria inferenza	$O(N \cdot d_k H)$ KV-cache	$O(N_{\text{state}})$ fissa	$O(N_{\text{state}})$
Campo recettivo	N (completo)	∞ (ma decadente)	$N_{\text{win}} + \infty$

Perché Mamba è cruciale per la genomica. Con sequenze di 1 Mbp e Transformer puro, la KV-cache occupa $10^6 \times d_k \times 2H \times 4$ byte ≈ 100 GB per layer. Mamba usa uno stato nascosto $h_t \in \mathbb{R}^N$ di dimensione $N \sim 64$–256, indipendente dalla lunghezza della sequenza: la memoria di inferenza è $O(N_{\text{state}})$ costante, rendendo pratico il processamento di genomi interi.

6.8 §507b Apprendimento contrastivo cross-modale per la biologia

L'approccio CLIP (Contrastive Language-Image Pre-training, Radford et al., 2021) ha dimostrato che l'allineamento di spazi di embedding da modalità diverse tramite perdita contrastiva produce rappresentazioni di alta qualità senza supervisione task-specifca. La stessa idea applicata alla biologia produce *BioClip*-like models.

6.8.1 Perdita InfoNCE per allineamento cross-modale biologico

Sia $(x_i^{(1)}, x_i^{(2)})$ una coppia di sequenze correlate (es. sequenza aminoacidica e struttura 3D della stessa proteina), e $(x_i^{(1)}, x_j^{(2)})$ con $i \neq j$ una coppia negativa. La *InfoNCE loss* (van den Oord et al., 2018) è:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{m}\sum_{i=1}^{m} \log \frac{e^{f(x_i^{(1)})^\top g(x_i^{(2)})/\tau}}{\sum_{j=1}^{m} e^{f(x_i^{(1)})^\top g(x_j^{(2)})/\tau}}, \tag{179}$$

dove f e g sono due encoder (es. PLM per la sequenza e una GNN equivariante per la struttura), e τ è la temperatura contrastiva.

Teorema 6.6 (InfoNCE come lower bound sulla mutua informazione)**.** *Massimizzare $-\mathcal{L}_{InfoNCE}$ su un batch di m campioni equivale a massimizzare un lower bound sulla mutua informazione $I(f(X^{(1)}); g(X^{(2)}))$:*

$$I(f(X^{(1)}); g(X^{(2)})) \geq \log m - \mathcal{L}_{InfoNCE}. \tag{180}$$

6.8.2 Applicazioni: ProteinCLIP e GenomeCLIP

- **ProteinCLIP**: allinea sequenza aminoacidica (encoder PLM) con struttura 3D (encoder GNN equivariante). Permette ricerca strutturale a partire da sequenza e viceversa, senza mai richiedere la predizione esplicita della struttura.
- **GenomeCLIP**: allinea sequenza genomica (encoder conv-stem) con profili di espressione (encoder RNA-seq). Le rappresentazioni apprese catturano la "firma funzionale" di regioni genomiche.
- **SequenceStructureFunction (S2F)**: allinea tutte e tre le modalità (sequenza + struttura + funzione) nella spirit di ESM-3, ma con training contrastivo invece di masked token prediction.

6.9 §509b Problemi aperti: la frontiera matematica dell'AI scientifica

6.9.1 1. Generalizzazione fuori-distribuzione in biologia strutturale

La distribuzione delle strutture proteiche nel PDB non è rappresentativa dello spazio di tutte le proteine naturali: le proteine de novo (non presenti in natura ma biologicamente plausibili) si trovano fuori dalla distribuzione di training. Il problema formale è:

Definizione 6.7 (Generalizzazione OOD per strutture proteiche)**.** Sia $\mathcal{P}_{\text{train}}$ la distribuzione delle proteine nel PDB e $\mathcal{P}_{\text{all}}$ la distribuzione di tutte le proteine biologicamente possibili. Un predittore strutturale f_θ generalizza OOD se:

$$\mathbb{E}_{x\sim\mathcal{P}_{\text{all}}}[\mathcal{L}(f_\theta(x), y^*(x))] \leq \mathbb{E}_{x\sim\mathcal{P}_{\text{train}}}[\mathcal{L}(f_\theta(x), y^*(x))] + \epsilon_{\text{OOD}}, \tag{181}$$

per un ϵ_{OOD} piccolo. La misurazione di ϵ_{OOD} è attualmente un problema aperto: non esiste un benchmark standard per la generalizzazione OOD in protein structure prediction.

6.9.2 2. Interpretabilità geometrica delle rappresentazioni latenti

Le rappresentazioni interne dell'Evoformer ($z_{ij}^{(k)}$ per layer k) codificano informazione geometrica sulle strutture proteiche. Il problema aperto è: *esiste un'analisi analoga alla mechanistic interpretability dei LLM per le feature geometriche di AlphaFold?* In particolare:

- Le singole dimensioni di z_{ij} corrispondono a invarianti geometrici interpretabili (distanze, angoli specifici)?
- Gli aggiornamenti triangolari apprendo "pattern triangolari" specifici (ad es., patch di β-sheet che soddisfano certe proporzioni triangolari)?

6.9.3 3. Leggi di scala con dati strutturali aumentati

Una questione aperta fondamentale è se le leggi di scala Chinchilla (22) si applicano all'AI strutturale con i dati generati da AlphaFold stesso ("strutture AF2"). Ci sono ora ~ 200 milioni di strutture AlphaFold nel database AFDB, più di $1000\times$ il PDB originale. La domanda è:

$$L(N, D_{\text{AFDB}}) \stackrel{?}{=} L(N, D_{\text{PDB}}) - \delta, \tag{182}$$

dove $\delta > 0$ è il guadagno dall'aggiunta di dati "sintetici". Se $\delta < 0$, le strutture AF2 introducono bias che peggiorano il modello. I risultati attuali (AF2 con distillation) suggeriscono $\delta > 0$, ma la dipendenza da N non è stata caratterizzata sistematicamente.

6.9.4 4. Convergenza cross-dominio: verso la matematica del biosistema unificato

Il problema aperto più ambizioso è la costruzione di un modello che mappi:

$$f : \text{Genoma} \to \text{Trascrittoma} \to \text{Proteoma} \to \text{Funzione cellulare}, \tag{183}$$

rispettando tutte le simmetrie rilevanti ad ogni livello:

- $\mathbb{Z}_2$ (complementarità WC) per il genoma;
- SE(3) per le strutture proteiche;
- Invarianza alle permutazioni di molecole nel stesso compartimento cellulare.

La funzione (183) è composizione di tre mappe con gruppi di simmetria *diversi*: non è chiaro come comporre gli inductive bias di simmetria attraverso livelli biologici diversi. Questo è probabilmente il problema matematicamente più profondo della biologia computazionale contemporanea.

A Appendice A — Glossario dei Simboli

Tabella 23: Glossario completo dei simboli usati nel Volume VIII

Simbolo	Significato
Q, K, V	Matrici query, key, value dell'attenzione
d_k	Dimensione delle proiezioni key/query
A	Matrice di attenzione (dopo softmax)
W^Q, W^K, W^V, W^O	Matrici di proiezione dell'attenzione
H	Numero di teste nell'attenzione multi-testa
N	Lunghezza della sequenza (token nel LLM)
T	Lunghezza della sequenza nel contesto autoregressivo
d	Dimensione dell'embedding
V	Dimensione del vocabolario
SE(3)	Gruppo speciale euclideo in $\mathbb{R}^3$: roto-traslazioni
SO(3)	Gruppo delle rotazioni proprie in $\mathbb{R}^3$
R	Matrice di rotazione $\in$ SO(3)
t	Vettore di traslazione $\in \mathbb{R}^3$
$T_i = (R_i, t_i)$	Frame rigido per il residuo i
S_N	Gruppo delle permutazioni di N elementi
$\mathbb{Z}_2$	Gruppo ciclico di ordine 2 (simmetria Watson-Crick)
s_i	Rappresentazione di sequenza del residuo i, $s_i \in \mathbb{R}^{c_s}$
z_{ij}	Rappresentazione di coppia (i, j), $z_{ij} \in \mathbb{R}^{c_z}$
M	Matrice MSA, $M \in \Sigma^{N_{\text{seq}} \times L}$
m_{si}	Embedding MSA per la sequenza s e la posizione i
$\varphi(\cdot)$	Operatore di complemento-inversione Watson-Crick
FAPE	Frame Aligned Point Error (loss geometrica, in Å)
pLDDT	Predicted Local Distance Difference Test (confidenza)
pTM	Predicted TM-score (confidenza globale)
L	Lunghezza della sequenza proteica (numero di residui)
N_{seq}	Numero di sequenze nell'MSA
c_s, c_z, c_m	Dimensioni delle rappresentazioni di sequenza, coppia, MSA
N, D	Numero di parametri del modello, numero di token di training
$\odot$	Prodotto elemento per elemento (Hadamard)
$\otimes$	Prodotto esterno
$\sigma(\cdot)$	Funzione sigmoid: $\sigma(x) = 1/(1 + e^{-x})$
softplus(x)	$\log(1 + e^x)$
r_ℓ	Risoluzione in bp al livello ℓ dello stem genomico
$\bar{\alpha}_t$	Coefficiente di noising cumulativo nel processo di diffusione
D_θ	Modello di denoising in AlphaFold 3
$\mathcal{I}(f)$	Indice di vincolo fisico (inductive bias)
$D^\ell(R)$	Matrice di Wigner di ordine ℓ per la rotazione R
Y_ℓ^m	Armonica sferica di ordine ℓ, grado m
PE(p)	Positional encoding alla posizione p
R_p	Matrice di rotazione RoPE alla posizione p
θ_i	Frequenza RoPE per la dimensione i
m_h	Pendenza ALiBi per la testa h
ε_c	Soglia di clamping nella FAPE (~ 10 Å)
LDDT$_i$	Local Distance Difference Test per il residuo i

Simbolo	Significato
$q = (q_0, q_1, q_2, q_3)$	Quaternione unitario per SO(3)
$\mathrm{sg}[\cdot]$	Stop-gradient nel VQ-VAE

B Appendice B — Note Storiche e Riferimenti

- **Vaswani et al. (2017)**: *Attention Is All You Need.* NeurIPS 2017. Introduce il Transformer. La formula dell'attenzione scalata (2) è letteralmente il titolo operazionalizzato.
- **Jumper et al. (2021)**: *Highly accurate protein structure prediction with AlphaFold.* Nature 596, 583–589. Il supplementary material (340 pagine) contiene la derivazione completa di tutti gli algoritmi presentati nella Sezione C.
- **Abramson et al. (2024)**: *Accurate structure prediction of biomolecular interactions with AlphaFold 3.* Nature 630, 493–500. Introduce il Pairformer e il modello di diffusione (§480).
- **Kaplan et al. (2020)**: *Scaling Laws for Neural Language Models.* OpenAI Technical Report. Introduce le equazioni (16).
- **Hoffmann et al. (2022)**: *Training Compute-Optimal Large Language Models* (Chinchilla). DeepMind. Derivazione della legge (22).
- **Su et al. (2024)**: *RoFormer: Enhanced Transformer with Rotary Position Embedding.* Neurocomputing 568. L'encoding RoPE (6) è oggi standard in LLaMA, Gemma, GPT-4.
- **Press et al. (2022)**: *Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation* (ALiBi). ICLR 2022.
- **Hayes et al. (2024)**: *Simulating 500 million years of evolution with a language model* (ESM-3). Science 385. Il primo SFM che unifica sequenza, struttura e funzione (§506).
- **Google DeepMind (2025)**: *AlphaGenome.* Modello per la predizione della funzione genomica multi-task. Preprint bioRxiv.
- **Kingma e Ba (2014)**: *Adam: A Method for Stochastic Optimization.* ICLR 2015. L'ottimizzatore (23)–(26) usato da tutti i sistemi discussi.
- **Dao et al. (2022, 2023)**: *FlashAttention* e *FlashAttention-2.* Meccanismo di attenzione memory-efficient descritto in §447.
- **Ainslie et al. (2023)**: *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints.* EMNLP 2023.
- **Arc Institute (2025)**: *Evo-2.* Modello a 40B parametri per sequenze biologiche, finestra di contesto 1 Mbp.
- **Avsec et al. (2021)**: *Effective gene expression prediction from sequence by integrating long-range interactions* (Enformer). Nature Methods 18.
- **Shazeer (2019)**: *Fast Transformer Decoding: One Write-Head is All You Need* (MQA). arXiv:1911.02150.

- **Loshchilov e Hutter (2019)**: *Decoupled Weight Decay Regularization* (AdamW). ICLR 2019. Equazione (27).
- **Pearl (1988)**: *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann. Fondamento teorico delle reti bayesiane (Vol. V, §276).

Silvano Salvador · *Matematica delle Reti Neurali* — Volume VIII · Aprile 2026

Continuazione di: Vol. I (§0–50) · Vol. II (§51–100) · Vol. III (§101–144)
Vol. IV (§201–246) · Vol. V (§276–341) · Vol. VI (§342–406) · Vol. VII (§407–446)

C Appendice C — Derivazioni Complete

C.1 C.1 Derivazione completa degli aggiornamenti triangolari dell'Evoformer

Questa appendice fornisce la derivazione completa dei quattro aggiornamenti triangolari che costituiscono il cuore dell'Evoformer.

C.1.1 Motivazione geometrica

La struttura proteica soddisfa la *disuguaglianza triangolare*: per tre residui i, j, k:

$$\|r_i - r_k\| \leq \|r_i - r_j\| + \|r_j - r_k\|. \tag{184}$$

Questa vincolo implica che la matrice di distanze D non sia arbitraria: non tutti gli aggiornamenti della coppia (i,k) sono coerenti con la fisica. Gli aggiornamenti triangolari implementano questo vincolo imponendo che ogni coppia (i,k) venga aggiornata aggregando l'informazione dalle coppie "intermedie" (i,j) e (j,k).

C.1.2 Aggiornamento moltiplicativo outgoing (forma completa)

L'aggiornamento della coppia (i,k) sommando su tutti i j che hanno $i \to j$ e $j \to k$:

$$g_{ij} = \sigma(W_g^{(1)} z_{ij} + b_g^{(1)}), \qquad \text{(gating scalare)} \tag{185}$$

$$a_{ij} = \text{LayerNorm}(z_{ij}) W_a^{(1)} + b_a^{(1)}, \tag{186}$$

$$b_{ij} = \text{LayerNorm}(z_{ij}) W_b^{(1)} + b_b^{(1)}, \tag{187}$$

$$z'_{ik} = g_{ik} \odot \text{LayerNorm}\left(\sum_{j=1}^{L} a_{ij} \odot b_{kj} \right) W_o^{(1)}, \tag{188}$$

dove $g_{ij} \in \mathbb{R}^{c_z}$ è il gate e $W_a, W_b \in \mathbb{R}^{c_z \times c_z}$, $W_o \in \mathbb{R}^{c_z \times c_z}$. Il termine $a_{ij} \odot b_{kj}$ è il prodotto elemento per elemento di due proiezioni lineari, e la somma su j aggrega l'informazione da tutti i "mediatori" j.

Interpretazione. La coppia (i,k) viene aggiornata guardando tutti i residui j come potenziali "intermediari": se sia z_{ij} che z_{kj} (la coppia *invertita* $j \to k$) contengono informazione rilevante, l'aggiornamento li combina moltiplicativamente (via il prodotto $a_{ij} \odot b_{kj}$, che è grande solo se entrambi sono grandi).

C.1.3 Aggiornamento moltiplicativo incoming

Simmetrico al precedente, ma aggrega via z_{ji} e z_{jk} (frecce in entrata a i e k dal mediatore j):

$$z''_{ik} = g^{(2)}_{ik} \odot \mathrm{LayerNorm}\left(\sum_{j=1}^{L} a_{ji} \odot b_{jk}\right) W^{(2)}_o. \tag{189}$$

C.1.4 Attenzione triangolare con starting node (forma completa)

$$q^{(h)}_{ij} = z_{ij} W^{(h)}_q, \quad k^{(h)}_{ij} = z_{ij} W^{(h)}_k, \quad v^{(h)}_{ij} = z_{ij} W^{(h)}_v, \tag{190}$$

$$b^{(h)}_{ij} = z_{ij} W^{(h)}_b, \qquad \text{(pair bias)} \tag{191}$$

$$a^{(h)}_{ijk} = \mathrm{softmax}_k\left(\frac{q^{(h)}_{ij} \cdot k^{(h)}_{ik}}{\sqrt{d_k}} + b^{(h)}_{jk}\right), \tag{192}$$

$$z'_{ij} = \sigma(W_g z_{ij}) \odot \mathrm{Concat}_h\left(\sum_k a^{(h)}_{ijk} v^{(h)}_{ik}\right) W_o. \tag{193}$$

Differenza con la MHA standard. Il termine $b^{(h)}_{jk}$ rappresenta una correzione di attenzione che dipende dalla coppia *target* (j,k): la query di i verso j è modulata dall'informazione della coppia (j,k) che media il legame triangolare. Questo non ha equivalente nella MHA linguistica.

C.1.5 Attenzione triangolare con ending node

Analoga a (193), ma con la somma su k nella dimensione delle *righe* della matrice di coppia:

$$a^{(h)}_{ijk} = \mathrm{softmax}_k\left(\frac{q^{(h)}_{kj} \cdot k^{(h)}_{ij}}{\sqrt{d_k}} + b^{(h)}_{ki}\right), \qquad z''_{ij} = \sigma(W_g z_{ij}) \odot \mathrm{Concat}_h\left(\sum_k a^{(h)}_{kij} v^{(h)}_{kj}\right) W_o. \tag{194}$$

C.2 C.2 Derivazione del gradiente di Adam con bias correction

Dimostriamo che il bias correction di Adam produce stime non distorte del gradiente.

Lemma C.1 (Valore atteso dei momenti non corretti)**.** *Sotto l'ipotesi che i gradienti siano i.i.d. con valore atteso $g^* = \mathbb{E}[g_t]$:*

$$\mathbb{E}[m_t] = g^*(1-\beta_1^t), \qquad \mathbb{E}[v_t] = \mathbb{E}[g_t^2](1-\beta_2^t).$$

Dimostrazione. Per m_t: si espande ricorsivamente $m_t = (1-\beta_1)\sum_{k=1}^{t} \beta_1^{t-k} g_k$. Prendendo l'aspettativa: $\mathbb{E}[m_t] = (1-\beta_1) g^* \sum_{k=1}^{t} \beta_1^{t-k} = g^*(1-\beta_1^t)$. La derivazione per v_t è analoga con β_2 e g_t^2. □ □

Dividendo per $(1-\beta_1^t)$ e $(1-\beta_2^t)$ rispettivamente, si ottengono stimatori non distorti $\hat{m}_t$ e $\hat{v}_t$.

Algorithm 4 Blocco Evoformer (singola iterazione)

Require: Rappresentazione sequenza $s \in \mathbb{R}^{L \times c_s}$, coppia $z \in \mathbb{R}^{L \times L \times c_z}$, MSA $m \in \mathbb{R}^{N_{\text{seq}} \times L \times c_m}$
Ensure: Rappresentazioni aggiornate s', z'
MSA row attention: $m \leftarrow m + \text{MSA-RowAttn}(m, z)$
MSA column attention: $m \leftarrow m + \text{MSA-ColAttn}(m)$
MSA transition: $m \leftarrow m + \text{FFN}(\text{LN}(m))$
Outer product mean: $z \leftarrow z + \text{OPM}(m)$ ▷ $O(L^2 N_{\text{seq}})$
Triangle mult. out.: $z \leftarrow z + \text{TriMultOut}(\text{LN}(z))$
Triangle mult. in.: $z \leftarrow z + \text{TriMultIn}(\text{LN}(z))$
Triangle attn. start: $z \leftarrow z + \text{TriAttnStart}(\text{LN}(z))$
Triangle attn. end: $z \leftarrow z + \text{TriAttnEnd}(\text{LN}(z))$
Pair transition: $z \leftarrow z + \text{FFN}(\text{LN}(z))$
Seq. from pair: $s \leftarrow s + \text{Linear}(\text{LN}(\text{ColAttn}(z)))$
return $s' = s,\ z' = z$

Algorithm 5 Invariant Point Attention (singolo blocco)

Require: Feature sequenza $s \in \mathbb{R}^{L \times c_s}$, coppia $z \in \mathbb{R}^{L \times L \times c_z}$, frames $\{T_i\}$
Ensure: Feature aggiornate s'
for ogni testa $h = 1, \ldots, H$ **do**
  $q_{i,h} \leftarrow s_i W_h^q,\ k_{i,h} \leftarrow s_i W_h^k,\ v_{i,h} \leftarrow s_i W_h^v$ ▷ Scalari
  $q_{i,hk}^{3D} \leftarrow T_i \cdot (s_i W_{hk}^{q,3D})$ per $k = 1, \ldots, K$ ▷ Punti 3D
  $k_{i,hk}^{3D} \leftarrow T_i \cdot (s_i W_{hk}^{k,3D})$ per $k = 1, \ldots, K$
  $v_{i,hk}^{3D} \leftarrow T_i \cdot (s_i W_{hk}^{v,3D})$ per $k = 1, \ldots, K$
  $a_{ij}^{(h)} \leftarrow \frac{q_i k_j}{\sqrt{c_s}} + b_{ij,h}^{\text{pair}} - \frac{\gamma_h}{2} \sum_k \| q_{i,hk}^{3D} - k_{j,hk}^{3D} \|^2$
  $A_{ij}^{(h)} \leftarrow \text{softmax}_j(a_{ij}^{(h)})$
  $o_i^{(h)} \leftarrow \sum_j A_{ij}^{(h)} v_{j,h}$ ▷ Parte scalare
  $o_i^{(h),3D} \leftarrow T_i^{-1} \sum_j A_{ij}^{(h)} v_{j,hk}^{3D}$ ▷ Parte 3D nel frame locale
end for
$s' \leftarrow s + \text{Linear}(\text{Concat}(o_i^{(h)}, o_i^{(h),3D}, \| o_i^{(h),3D} \|^2)_{h=1}^{H})$
return s'

Algorithm 6 Generazione autoregressiva con KV-cache

Require: Prompt $(x_1, \ldots, x_p)$, lunghezza massima T
Ensure: Sequenza generata $(x_{p+1}, \ldots, x_{p+T})$
Processa prompt: calcola e memorizza $(K_1, V_1), \ldots, (K_p, V_p)$
$t \leftarrow p$
while $t < p + T$ e $x_t \neq \text{EOS}$ **do**
  Calcola (K_t, V_t) per il nuovo token x_t
  Aggiorna cache: $\mathcal{K} \leftarrow \mathcal{K} \cup \{K_t\},\ \mathcal{V} \leftarrow \mathcal{V} \cup \{V_t\}$
  $h_t \leftarrow \text{MHA}(Q = s_t W^Q,\ K = \mathcal{K},\ V = \mathcal{V})$
  $\ell_t \leftarrow W_{\text{vocab}}\, h_t$ ▷ Logit sul vocabolario
  $x_{t+1} \sim \text{Categorical}(\text{softmax}(\ell_t / \tau))$ ▷ Sampling con temperatura τ
  $t \leftarrow t + 1$
end while

D Appendice D — Raccolta degli Algoritmi

D.1 D.1 Blocco Evoformer completo

D.2 D.2 Algoritmo IPA completo

D.3 D.3 Generazione autoregressiva con KV-cache

E Appendice E — Schede Riassuntive dei Modelli

Tabella 24: Scheda tecnica: AlphaFold 2 (Jumper et al., 2021)

Parametro	Valore
Parametri totali	$\sim$ 93M
Profondità Evoformer	48 blocchi
Profondità Structure Module	8 blocchi
Dimensioni (c_s, c_z, c_m)	$(384, 128, 256)$
Teste IPA (H, K)	$(12, 4)$
Ottimizzatore	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Batch size	256 (aggiornato con gradient accumulation)
Dataset	PDB ($\sim$130k strutture) + distillazione
Training time	$\sim$ 11 settimane su $\sim$128 TPU v3
Performance (CASP14)	TM-score > 0.9 per 60% dei target

Tabella 25: Scheda tecnica: AlphaFold 3 (Abramson et al., 2024)

Parametro	Valore
Parametri totali	$\sim$ 700M
Profondità Pairformer	48 blocchi
Modello di diffusione	36 blocchi, condizionato su Pairformer
Dominio	Proteine, DNA, RNA, ligandi, modificazioni
Ottimizzatore	AdamW
Dataset	PDB + CSD (strutture cristalline di piccole molecole)
Performance (proteine)	Superiore ad AF2 su corpi flessibili e complessi
Novità architetturale	Pairformer sostituisce Evoformer; diffusione sostituisce IPA

F Appendice F — Geometria Differenziale di SE(3)

F.1 F.1 Struttura di gruppo di Lie

SE(3) è un gruppo di Lie di dimensione 6. Come gruppo di Lie, è una varietà differenziabile i cui prodotti e inversioni sono mappe lisce. L'elemento neutro è $(I, 0) \in \mathrm{SO}(3) \times \mathbb{R}^3$.

F.1.1 La mappa esponenziale

La mappa esponenziale $\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$ manda un elemento dell'algebra di Lie (tangente in $(I, 0)$) a un elemento del gruppo:

$$\exp \begin{pmatrix} [v]_\times & u \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} e^{[v]_\times} & Vu \\ 0 & 1 \end{pmatrix}, \tag{195}$$

Tabella 26: Scheda tecnica: ESM-3 (Hayes et al., 2024)

Parametro	**Valore**
Parametri (modello principale)	98M, 300M, 3B (tre scale)
Modalità	Sequenza + struttura (VQ-VAE) + funzione (GO terms)
Obiettivo	Masked token prediction (BERT-style)
Codebook strutturale	$K = 4096$ centroidi, risoluzione ~ 0.5 Å
Dataset	771M sequenze proteiche + 236M strutture
Performance	Genera proteine fluorescenti funzionalmente nuove
Novità	Unifica PLM e predizione strutturale in un unico modello

Tabella 27: Scheda tecnica: AlphaGenome (Google DeepMind, 2025)

Parametro	**Valore**
Finestra genomica	1 Mbp
Risoluzioni	16 bp, 128 bp, 1 kbp, 8 kbp
Task predetti	> 50 (RNA-seq, ATAC-seq, ChIP-seq, Hi-C, splicing)
Ottimizzatore	AdamW con gradient clipping
Dataset	Genoma umano hg38, dati funzionali ENCODE, GEO
Simmetria	Complementarità Watson-Crick ($\mathbb{Z}_2$)
Confronto	Supera Enformer di $\sim 12\%$ correlazione su RNA-seq

dove $[v]_\times$ è la matrice antisimmetrica di $v \in \mathbb{R}^3$:

$$[v]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}, \tag{196}$$

$e^{[v]_\times}$ è la formula di Rodrigues:

$$e^{[v]_\times} = I + \frac{\sin\|v\|}{\|v\|}[v]_\times + \frac{1-\cos\|v\|}{\|v\|^2}[v]_\times^2, \tag{197}$$

e la matrice V è:

$$V = I + \frac{1-\cos\|v\|}{\|v\|^2}[v]_\times + \frac{\|v\| - \sin\|v\|}{\|v\|^3}[v]_\times^2. \tag{198}$$

Importanza per AlphaFold. La mappa esponenziale permette di lavorare *linearmente* nello spazio tangente $\mathfrak{se}(3)$, applicando la mappa esponenziale alla fine. Gli aggiornamenti incrementali dei frames (79) possono essere interpretati come passi nel tangente space: dato uno step $(\delta v, \delta t) \in \mathfrak{se}(3)$, il frame aggiornato è $T' = T \cdot \exp(\delta v, \delta t)$. Questo è più stabile numericamente dell'aggiornamento diretto delle matrici di rotazione.

F.1.2 Geodetiche e distanza di Riemanniana

La *distanza di Riemanniana* tra due elementi $T_1, T_2 \in \mathrm{SE}(3)$ (con la metrica biinvariante) è:

$$d(T_1, T_2) = \|\log(T_1^{-1}T_2)\|_{\mathfrak{se}(3)}, \tag{199}$$

dove $\log : \mathrm{SE}(3) \to \mathfrak{se}(3)$ è l'inversa della mappa esponenziale. Per rotazioni pure:

$$d_{\mathrm{SO}(3)}(R_1, R_2) = \|\log(R_1^\mathsf{T} R_2)\|_F = \arccos\left(\frac{\mathrm{tr}(R_1^\mathsf{T} R_2) - 1}{2}\right), \tag{200}$$

che per angoli di rotazione piccoli è approssimativamente $\|\delta R\|_F$ dove $\delta R = R_1^\top R_2 - I$.

F.2 F.2 Interpolazione SLERP per rotazioni

La *Spherical Linear Interpolation* (SLERP) interpola tra due quaternioni unitari q_0, q_1 lungo la geodetica su S^3:

$$\mathrm{SLERP}(q_0, q_1; t) = q_0 \cdot (q_0^{-1} q_1)^t, \quad t \in [0,1], \tag{201}$$

dove $(q)^t = e^{t \log q}$ con $\log q = \frac{\arccos(q_0)}{|\sin(\arccos(q_0))|}(q_1, q_2, q_3)$ (parte vettoriale normalizzata moltiplicata per l'angolo).

SLERP è cruciale nel design di AlphaFold 3: durante la diffusione (93), il rumore viene aggiunto alle rotazioni tramite SLERP invece della somma Euclide (che violerebbe $\|q\| = 1$). Il processo di noising su SO(3) diventa:

$$q_t = \mathrm{SLERP}(q_0, q_\epsilon; \sqrt{1 - \bar{\alpha}_t}), \quad q_\epsilon \sim \mathrm{Uniform}(\mathrm{SO}(3)). \tag{202}$$

F.3 F.3 Misura di Haar su SO(3)

La misura di Haar μ_H su SO(3) (la misura invariante alla rotazione) ha volume totale:

$$\mu_H(\mathrm{SO}(3)) = \int_{\mathrm{SO}(3)} \mathrm{d}R = 8\pi^2. \tag{203}$$

Questa quantità appare nel bound di generalizzazione (154): il termine di riduzione della complessità per $G = \mathrm{SO}(3)$ è $-\log(8\pi^2)/m$, quantificando il guadagno dall'invarianza SE(3) in termini di campioni di training effettivi.

G Appendice G — Connessioni con la Teoria delle Informazioni Biologica

G.1 G.1 Capacità dei canali biologici

La codifica dell'informazione in biologia può essere formalizzata tramite la teoria dei canali di Shannon. Il canale DNA→Proteina è:

- Input: codone (tripletta di nucleotidi), alfabeto $4^3 = 64$ simboli;
- Output: aminoacido, alfabeto 20 simboli;
- Capacità del canale: $H(\text{codone}) = \log_2(64) = 6$ bit per posizione;
- Ridondanza: la degenerazione del codice genetico (64 codoni per 20 AA + stop) introduce ridondanza di $\approx 6 - \log_2(20) \approx 1.7$ bit per posizione.

Il codice genetico non è casuale: la ridondanza è distribuita in modo da minimizzare l'effetto delle mutazioni puntiformi. Mutazioni al terzo codon position ("wobble") lasciano spesso invariato l'aminoacido, mentre mutazioni al primo o secondo possono cambiarlo.

G.2 G.2 Entropia evolutiva e MSA

L'MSA di AlphaFold può essere vista come una stima del canale evolutivo: ogni riga è un campione di una distribuzione di sequenze condizionata alla struttura 3D (ipotesi di Anfinsen). L'informazione mutua tra le colonne i e j dell'MSA:

$$I(M_i; M_j) = H(M_i) + H(M_j) - H(M_i, M_j) \geq 0, \tag{204}$$

misura la co-evoluzione tra i residui i e j. Il metodo *Direct Coupling Analysis* (DCA) fattorizza la distribuzione dell'MSA come:

$$\log p(m_{1:L}) \approx \sum_i h_i(m_i) + \sum_{i<j} J_{ij}(m_i, m_j), \tag{205}$$

dove J_{ij} sono i "coupling". I picchi di $\|J_{ij}\|$ corrispondono a contatti strutturali. AlphaFold "apprende" implicitamente questi coupling attraverso l'attenzione sull'MSA, senza calcolarli esplicitamente.

G.3 G.3 La compressione dell'informazione strutturale

La struttura di una proteina in PDB viene salvata tipicamente come file di coordinate $\sim$ 50–500 kB. AlphaFold la predice a partire da una sequenza di $\sim$ 1–5 kB. La predizione strutturale è quindi un problema di *decompressione*: estrarre $\sim 100\times$ più informazione di quella contenuta nella sequenza pura. Questo è possibile perché:

1. Le strutture proteiche obbediscono a vincoli fisici che riducono il numero di configurazioni ammissibili (piano di Ramachandran, vincoli sui legami);
2. L'evoluzione ha selezionato sequenze che si piegano in strutture stabili, rendendo lo spazio delle strutture accessibili molto più piccolo di $\mathbb{R}^{3L}$.

La "quantità di informazione" aggiuntiva fornita dall'evoluzione e dalla fisica è esattamente l'inductive bias $\mathcal{I}(f)$ del §505 e §508 — formalizzata qui come capacità di decompressione biologica.

Riferimenti bibliografici

[1] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[2] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Gymer, D. Hassabis, R. Hewtson, T. Hhorne, M. Langner, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024.

[3] T. Hayes, R. Rao, H. Akin, N. J. Sofroniew, D. Oktay, Z. Lin, R. Verkuil, V. Q. Tran, J. Deaton, M. Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, 385(6707):eadk9777, 2024.

[4] Google DeepMind. AlphaGenome: Advancing genomic sequence modelling. Technical report, Google DeepMind, 2025.

[5] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 16344–16359, 2022.

[6] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[8] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskiy, T. Cai, D. de Las Casas, A. Bornschein, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 30016–30030, 2022.

[9] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[10] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[11] N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 4596–4604, 2018.

[12] A. Kosson, A. Vyas, M. Bhatt, and M. Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. *arXiv preprint arXiv:2402.12192*, 2024.

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[14] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling. SE(3)-transformers: 3D roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[15] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6840–6851, 2020.

[17] Ž. Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203, 2021.

[18] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.